2007-12

Discrete-event simulation with agents for modeling of dynamic asymmetric threats in maritime security

Ng, Chee Wan.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DISCRETE-EVENT SIMULATION WITH AGENTS FOR MODELING OF DYNAMIC ASYMMETRIC THREATS IN MARITIME SECURITY

by

Chee Wan Ng

December 2007

Thesis Advisor:	Arnold H. Buss
Co-Advisor:	John Hiles

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Discrete-Event Simulation with Agents for Modeling of Dynamic Asymmetric Threats in Maritime Security			**5. FUNDING NUMBERS**
6. AUTHOR(S) Chee Wan, Ng			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Maritime security has become an important security focus area, due to the impact that piracy and terrorism have on the global economy. There are many studies on detecting and engaging asymmetric threats in ports and waterways. However, the threats are typically modeled too simply, with predefined or random paths and fixed responses. There is a need to model representing dynamic, asymmetric threat behaviors so that future threat-response models will be a more realistic evaluation against a dynamically adaptive foe.

Discrete-event simulation (DES) was used to simulate a typical port-security, local, waterside-threat response model and to test the adaptive response of asymmetric threats in reaction to port-security procedures, while a multi-agent system (MAS) was used to provide the complex adaptive behaviors for our threats. Cover and dynamic pathfinding were used with the sensor framework in Simkit to enhance the spatial interactivity of the agents.

This study found that MAS asymmetric threats demonstrate greater flexibility of behaviors and show potential for adaptability. These dynamic asymmetric threats will enable simulation of a wider variety of maritime-threat scenarios, and play an important part in improving the plans for future maritime force and infrastructure configurations.

14. SUBJECT TERMS Discrete-Event Simulation, Multi-agent System, Asymmetric Threat, Piracy, Terrorism, Maritime Security, Port Security			**15. NUMBER OF PAGES** 99
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DISCRETE-EVENT SIMULATION WITH AGENTS FOR MODELING OF DYNAMIC ASYMMETRIC THREATS IN MARITIME SECURITY

Chee Wan Ng
Civilian, Defence Science and Technology Agency, Singapore
M.Sc. (Elect. Eng.), National University of Singapore, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
December 2007

Author: Chee Wan Ng

Approved by: Arnold H. Buss
Thesis Advisor

John Hiles
Co-Advisor

Rudy Darken
Chairman, Department of MOVES

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Maritime security has become an important security focus area, due to the impact that piracy and terrorism have on the global economy. There are many studies on detecting and engaging asymmetric threats in ports and waterways. However, the threats are typically modeled too simply, with predefined or random paths and fixed responses. There is a need to model representing dynamic, asymmetric threat behaviors so that future threat-response models will be a more realistic evaluation against a dynamically adaptive foe.

Discrete-event simulation (DES) was used to simulate a typical port-security, local, waterside-threat response model and to test the adaptive response of asymmetric threats in reaction to port-security procedures, while a multi-agent system (MAS) was used to provide the complex adaptive behaviors for our threats. Cover and dynamic pathfinding were used with the sensor framework in Simkit to enhance the spatial interactivity of the agents.

This study found that MAS asymmetric threats demonstrate greater flexibility of behaviors and show potential for adaptability. These dynamic asymmetric threats will enable simulation of a wider variety of maritime-threat scenarios, and play an important part in improving the plans for future maritime force and infrastructure configurations.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professor Arnold Buss for his expert guidance, heartfelt encouragement, and unwavering patience. Modeling and simulation is an arcane field for someone new to the field, and his advice helped resolve many mysteries.

Multi-agent systems have inherited a long history from biology, and Professor John Hiles has shown me how to appreciate both. I would like thank him for providing inspiration and guidance for this work.

I also thank Professor Chris Darken for his lectures, which motivated the pathfinding in this work, and Professor Tony Ciavarelli and Mr. Curtis L. Blais for their guidance and support during the course of this research.

“Port Security Strategy 2012” [3] provided a solid start for this thesis. I would like to congratulate our Systems Engineering and Analysis Cohort 11 team for their good systems-engineering work. In particular, I thank LT Morgan Ames, Mr. Dennis Lim Thiow Yong, Mr. Jeffrey Chan Chun Man, Mr. Ng Chee Wai, and Mr. Koh Kim Leng for being such great teammates. In addition, I would like to pay special thanks to Mr. Koh Kim Leng for his close partnership in our software development of the port security 2012 simulation.

Most importantly, I would also like to thank my wife Caryn for her patience, understanding, and support on our journey at the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. INTERNATIONAL MARITIME SECURITY

In this age of global trade, millions of dollars in goods and supplies are being shipped across the world at any moment. Any disruption in this global supply chain will affect the economies of many countries. All countries in this global economy have a vested interest in ensuring the protection of maritime activities. [1] [2]

Corresponding to the increase in trade, there is an increase in human activity in port areas to support the increase in shipping. With the growing affluence of nations, there is also an increase in leisure cruising and marina use. From the perspective of the would-be terrorist, these popular activities offer an attractive political target for achieving the objective of instilling fear into the daily lives of a target nation's people. [9]

International maritime security is essential to protect and secure these commercial shipping and human recreational activities in port areas and international waters.

B. MARITIME ASYMMETRIC THREATS

Asymmetric threats are thriving in the maritime environment. While lacking in big guns and the latest expensive modern technology for a major battle, these threats make use of simple equipment easily accessible in the open market, in conjunction with operational tactics designed to exploit the weaknesses of more advanced and expensive technologies. The operational tactics of the asymmetric threat are employed against established navies and other formal security establishments by terrorists, insurgent groups, and pirates. Several specific threats will now be described.

The number-one threat in most American minds is Al-Qaeda. The United States' Department of Defense defines Al-Qaeda as "A radical Sunni Muslim umbrella organization established to recruit young Muslims into the Afghani mujahideen and is aimed to establish Islamist states throughout the world, overthrow 'un-Islamic regimes', expel U.S. soldiers and Western influence from the Gulf, and capture Jerusalem as a Muslim city" [9]. Using small suicide crafts, Al-Qaeda attacked the USS *Cole*, a destroyer-class ship, in the Yemeni port of Aden on October 12, 2000 [15] [16] [17] and the French tanker *Limburg* in the Gulf of Aden on October 6, 2002 [18] [19].

Another example of a deadly threat is the Liberation Tigers of Tamil Eelam (LTTE), a rebel group that has been fighting for an independent Tamil homeland in the north of Sri Lanka since 1976. The naval wing of the LTTE, the Sea Tigers, demonstrate the highest level of naval organization, tactics, technology, and power of any insurgent group to date, and has destroyed numerous boats, even a warship in the Sri Lankan navy (SLN) [13].

The Abu Sayyaf group (ASG) is an example of the several militant Islamist separatist groups fighting for an independent Islamic state in western Mindanao and the Sulu Archipelago, with the stated goal of creating a pan-Islamic superstate across southeast Asia [9]. The ASG sank the Philippine Super Ferry 14 off Manila on February 26, 2004 [13].

While not part of any terrorist organization, pirate groups target commercial and civilian ships for robbing of cash, belongings, and navigational equipment, hijacking of cargoes and vessels, and kidnapping for ransom. Sometimes the crew is killed. [14] [12]

C. TACTICS OF MARITIME ASYMMETRIC THREATS

Maritime asymmetric threats employ a combination of tactics, such as fast speed, innocent speed, legal cover, camouflage, deception, and reducing radar detection signature. This study focus is on navigational tactics. It is assumed that

once a threat gets close to its target, it will carry out its mission successfully. The navigational tactics of the maritime asymmetric threats are categorized as follows.

1. Outrunning

The Sea Tigers's fast-attack craft can chase at 40 to 45 knots, but must cut speed to 20 knots to fire weapons accurately [13].

Pirate groups in Southeast Asia have also demonstrated an ability to capture nonmilitary ships and escape and evade capture during the course of operations [13].

2. Maintaining Innocent Speed

Al Qaeda used deception to get close to the USS *Cole:* a small boat, mixing with the harbor's refueling crafts, was likely to have moved slowly without giving away its intent until the last moment, when it headed directly towards the *Cole* [16].

3. Following a Ship

On October 23, 2000, five Sea Tigers suicide crafts followed a regular cargo vessel into Trincomalee harbor; two were destroyed, two escaped, and one reached an SLN-operated ferry, the A541. The A541 was crippled and forty SLN sailors injured. The successful suicide craft was aided in its final approach by diversionary mortar and rocket fire from a land-based LTTE unit [13].

4. Hiding between Ships

On January 7, 2006, a Sea Tigers suicide craft hid inside a cluster of fishing vessels in Trincomalee harbor during the night and waited for an SLN patrol boat to pass before emerging and ramming it. Fifteen SLN sailors were killed [13].

5. Swarming

On May 1, 2006, the Sea Tigers mounted a swarm attack just outside the breakwater of Trincomalee harbor, using five attack crafts to open fire on a single SLN Dvora patrol boat. One Sea Tigers craft was sunk, while the *Dvora* was damaged and ten SLN sailors killed. Due to command-and-control limitations, each group of Sea Tigers attack crafts consists of two to three crafts, but they can operate up to 80km apart and use speed to concentrate their forces when necessary [13].

D. APPROACH

As part of a team performing an earlier study on “Port Security 2012” [1], a simulation of port-security measures was jointly developed against small-boat threats. The main focus was on modeling and simulation the performance of port-security measures, including radars, thermal-vision sensors, sonars, patrol crafts, helicopters, unmanned surface vehicles etc. The small-boat threats were assigned routes randomly chosen from a table of fixed routes of attack. While sufficient for a broad study on the relative effectiveness of various port-security measures, improved fidelity in the simulation of battlefield entities was deemed necessary for deeper exploration of the operational tactics applicable in the engagement environment.

Further study reveals that the emphasis of such extended efforts [4] [5] [6] [7] is typically on security measures, while simplifying the capability, and especially the adaptability, of asymmetric threats. Real-world incidents have proven different—asymmetric threats are highly agile, able to adapt their tactics quickly to changes in the defense infrastructure, and on the strength of such capability, likely to exploit any known or hitherto-unknown weakness of a defense infrastructure [8]. There is a need to simulate the adaptability of asymmetric threats, to better explore the operational tactics applicable in the engagement environment, and to reveal operational weaknesses that can emerge in the aftermath of changes to the defense infrastructure.

Multi-agent systems (MASs) in simulations are known for their ability to adapt to and explore their environment. It is hypothesized that a MAS simulation will provide a mechanism for simulating the adaptability of asymmetric threats and, in so doing, reveal operational tactics applicable in the engagement environment.

Discrete-event simulation (DES) provides the foundation for this port-security simulation. It allows for true time–space simulation and is free from the time truncation encountered in time-period based simulations. This is especially important for military simulations, in which time and spatial dimensions play essential roles. DES is also an event-driven simulation: loosely coupled events perform actions, trigger other events, and drive the simulation. Its methodology corresponds closely to the human concept of events and actions and allows for natural encoding of the human-defined rules and behaviors in a scenario. DES and agent-based technology are a natural fit, because of DES's encoding and application of agent rules and behaviors.

There is a need to explore how an asymmetric threat exploits weaknesses in changes made to a defense infrastructure. Discrete-event simulation can provide a foundation for simulating defense infrastructures, while agent-based simulation of the threats helps explore their possible range of actions.

It is noted that this study is only a small part of the multifaceted effort towards combating terrorism, which includes technological, informational, economical, social, and psychological approaches.

E. OBJECTIVE AND SCOPE OF THIS STUDY

The present objective is to identify and evaluate the adaptability of MAS DES asymmetric threats in defeating maritime-security procedures.

This study asks the following questions:

1) How can adaptable MAS asymmetric threats be encoded into DES models of maritime security?
2) Does the MAS DES model demonstrate increased adaptability?

3) Can adaptable MAS DES models of asymmetric threats increase the success rate at defeating maritime-security procedures, compared to non-adaptable DES model of asymmetric threats?

F. RELATED WORK

At the Naval Postgraduate School, master's students are exploring agent-based technologies in maritime security and other environments. Terence Tan is studying the application of conceptual-blending theory to agents, for naval tactical-plan generation in littoral-water operation [20], and Ryan Tan is studying the application of agents to modeling of human behavior, in diverse areas such as logistic behaviors in riverine operations and social attendance [21].

II. THEORY BACKGROUND

A. INTRODUCTION

This chapter provides background on the DES methodology used to model a port-security environment, a description of MAS methodologies used in this study, and background on the integration of DES and MAS. The concept of adaptability in a simulation, and how it can be verified and validated, is also discussed.

B. DES FOR PORT SECURITY

1. DES

Law describes DES as “modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time” [22]. These points in time are the instants at which an event occurs, and nothing happens in the time between them. Law says that DES can be implemented using next-event time advancement or fixed-increment time advancement, and that next-event time advancement is the approach used by all major simulation software. This is because fixed-increment advancement has well known problems of time truncation, introducing errors that affect the accuracy of simulation output and the ability to decide which events come first when simultaneous events occur. It also suffers inherently from wasted time increments doing nothing.

Next-event time advancement turns the simulation clock to the time when the next event occurs, and in this way, applies a continuous time dimension. It must be noted that there are scenarios in which fixed time increments do not suffer from inaccuracy, incorrectness and inefficiency, but only where events occur at fixed times—which rarely applies in dynamic scenarios such as military simulations.

In addition to truncation of time, truncation of space can also be undesirable in simulations involving the spatial dimension. Depending on the fidelity of the scenario, its inputs, and desired results, it would be inaccurate and incorrect to truncate spatial values beyond the required resolution. Because the spatial dimension plays an important role in military simulations, truncation should be avoided where possible, while weighing other factors such as performance and resource constraints.

DES can be described using event graphs. An event graph consists of event nodes and scheduling edges. The edges may be scheduled only if a condition is satisfied. The next event may be scheduled immediately or only after a specified time. The figure below shows event nodes A and B, one scheduling edge from A to B, one condition, i, and a scheduling time, t. [23]

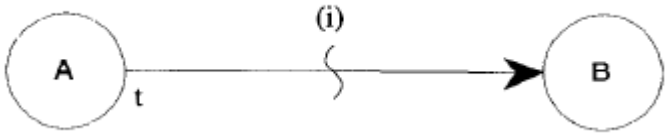


Figure 1 Fundamental Simulation Graph Construct. Whenever event A occurs, if condition i is true after A's state transition, event B is scheduled to occur t time units later (From [23])

An event node can also cancel another previously scheduled event. This canceling edge is shown in the following figure as a dotted line.

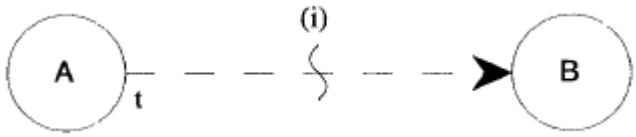


Figure 2 A Canceling Edge (From [23])

An event node can pass attributes on edges to an event node taking in a parameter. As shown in the figure below, the attribute k is passed with the canceling edge to event node B(j). Attribute k is passed in as parameter j [24].

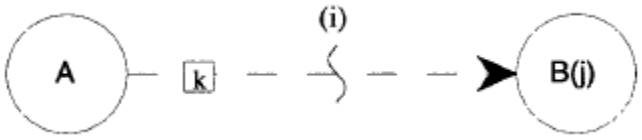


Figure 3 A Canceling Edge with An Attribute Passed to Event Node B (From [24])

As seen in the following figure, the changes of the state variables can also be depicted below the event node.

Figure 4 Simulation Graph for Poisson Process (From [23])

Event-graph models have been shown capable of modeling any system that can be implemented on a computer. Thus, they can represent existing complex systems as well as future complex systems that might be implemented in other fashions. [25]

In the context of this paper, DES is defined as using next-event time advance and a continuous spatial dimension and described using event graphs.

In the real world of limited time and budgets, it is important to be able to know and manage the effort and resources required for a proposed simulation study using discrete-event simulation. Methods to measure the complexity of event-graph models have been introduced, including vertex count, edge-to-vertex ratio, cyclomatic number (number of control paths), size of event lists, and

combinations of these methods [26]. These can help scope, prioritize and bring focus to a simulation project and ensure that it is successfully executed within budget.

It may be difficult to eliminate complexity completely, but complexity can be reduced in discrete-event simulation without reducing functionality, by simplifying the event graphs through various methods, including application of hierarchical event graphs [27], loosely coupled, component-based modeling with design patterns [28], and listener event-graph objects (LEGOs) [29].

2. Simkit

First published in 2001 [30], Simkit is a free and open-source discrete-event-simulation Java library made available under the GNU lesser general-public license (LGPL) [31]. This allows Simkit to be used in proprietary programs, as opposed to the GNU general-public license (GPL), which requires the program to be offered free [33]. This proprietary accommodation is important to the commercial and military sectors. Simkit is used for teaching discrete-event simulation at the Naval Postgraduate School (NPS) and as the foundation library for Viskit, Diskit, and Gridkit, which are visual modeling, distributed interactive simulation (DIS), and cluster-project initiatives for discrete-event simulation at NPS. NPS researchers have used Simkit to create discrete-event simulations for army, air, navy, maritime, and other scenarios.

Simkit implements discrete-event simulation with LEGOs, using loosely coupled, component-based modeling with design patterns such as listener, mediator, referee, and factory [34]. Present work by Koh refines the modeling approaches in Simkit using design patterns [32].

Simkit implements events, scheduling edges, a future event-list scheduler, cancellation of events, and parameter passing on scheduling edges to enable discrete-event simulation. Simkit also implements random variates, including

Bernoulli, binomial, exponential, geometric, normal, Poisson, uniform, Weibull and many others, for the application of probability distribution in the models. [30]

Simkit uses two listener patterns, “SimEventListener” and “PropertyChangeListener”, to enable loosely coupled, component-based modeling [30]. “SimEventListener” allows for dividing a large event graph into components and enables loose coupling between them. “PropertyChangeListener” allows for the decoupling of statistics-data collection from events, which also simplifies the programming of event logic.

While Simkit can be used for general programming of any type of DES model, it also provides a framework for simple movement and detection in discrete-event simulation [35]. A referee is used to compute when and where a target is detected by a sensor, based on the target’s movement and the sensor’s specifications. This schedules an event to the mediator, which decides the conditions under which the sensor will be notified that a target is detected. These sensor and movement functionalities are important for the simulation of models with a spatial dimension, and these basic capabilities in Simkit are a useful foundation for simulating the many sensors and movement platforms found in military command and control. This is an area where Simkit excels in comparison with commercially available, general-purpose DES products such as Arena [36] and Extend [37], etc.

3. A* Pathfinding in Simkit

A* search allows for efficient finding of the optimal path from a starting point to a destination over a graph map [38] [39]. There are many variants of the A* search and they are commonly used in game engines in which intelligent bots have to navigate their way around a map. Intelligent pathfinding is important in this project because it makes it possible to simulate the dynamic movement of small-boat threats as they respond to inputs from their environment. A* search is also used to enable patrol crafts to chase the small boat threats without colliding into the land mass.

Sullivan implemented an A* search in Diskit to allow pathfinding between the centers of rectangular zones [7]. To enhance continuity of space and allow dynamic movements, a higher resolution A* pathfinding map that completely covers the sea mass was generated to utilize the generic A* search in Simkit\Diskit created by Professor Arnold H. Buss.

4. Relevant Work

In addition to creating models for army, airforce and navy, Simkit has been used at NPS to create maritime scenarios. Childs's thesis explored the creation of a waterfront force-protection simulation using Simkit [4]. Sullivan's thesis demonstrated the application of real-world graphical and physical models to the waterside-security simulation using Simkit, Diskit, Viskit and Savage Studio [7]. The "Port Security Strategy 2012" project by the Systems Engineering and Analysis Cohort 11 of NPS's Meyer Institute of Systems Engineering simulated multiple port-security measures, using Simkit to compare their performance in identifying small-boat threats. Entities simulated in the port security environment include a patrol helicopter, patrol crafts, radar sensors, small-boat threats, big ships, and small boats [1].

In this thesis, the DES model in "Port Security Strategy 2012" is extended to simulate the engagement of patrol crafts and a patrol helicopter with small-boat threats, while interacting with the other entities in the environment.

C. AGENTS FOR PORT SECURITY

In their description of a taxonomy for agents, Franklin and Graessaer defined an autonomous agent as "a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future". Autonomous agents can be characterized by the following properties: reactive, sensing and acting; autonomous, goal-oriented, proactive and purposeful; temporally continuous; communicative and socially able; able to learn and adapt; mobile;

and flexible in their actions and character. [43] In this study, it is hypothesized that agents can improve the adaptability of small-boat threats as they navigate to and attack a high-value zone in a port security environment. This hypothesis will be validated if the agents representing the small-boat threats do demonstrate a greater variety of navigation in attacking the high-value zone in the port-security environment.

1. Multi-Agent System (MAS)

A MAS can consist of many subagents constituting a single agent or a society of agents [43].

In a multiple-subagent system, each subagent can represent a different task component of a single agent, and each component is able to sense and react to the environment [43]. For example, a patrol craft can have one agent for sensing and reacting to other agents, another agent for scanning high-value zones, and another for performing intercepts.

In a society of agents, multiple agents can interact and optionally communicate with each other [43]. For example, an agent representing a patrol craft can search an area for small-boat threats and broadcast detections to the command center. An agent representing the command center can receive broadcasts from multiple patrol agents and coordinate their interception of a small-boat threat. The agent representing the threat can maneuver to avoid interception while still pursuing its goal of reaching a high-value zone.

The following diagram shows a model of an agent in a MAS. The agent has a mental model, an input suite taking a sensing stream from the environment, and an output suite taking actions that affect the environment [44].

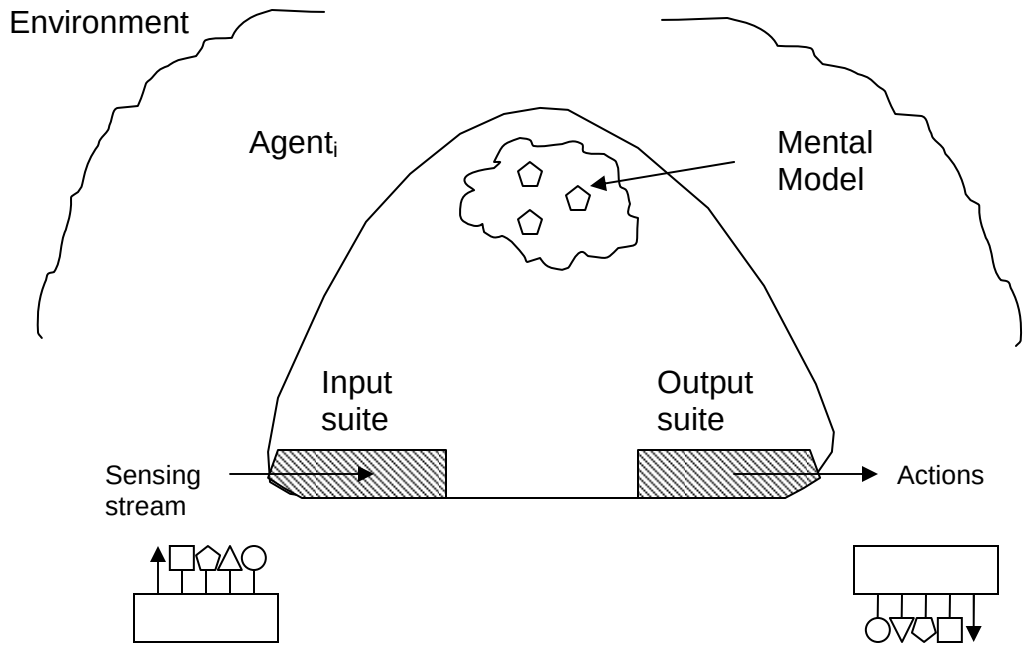


Figure 5 Model of an Agent in a MAS (From [44])

Each agent has a set of goals, conditions, methods, and rules, as shown in the figure below [44]. The goals identify the agent's objectives. The conditions chose which subsequent decision to carry out. The methods process the sensing stream into the input suite to update the weights in the mental model of the agent, and this affects its subsequent decision. Rules consist of rule conditions and actions. Rule conditions specify whether actions can be legally performed in an environment.

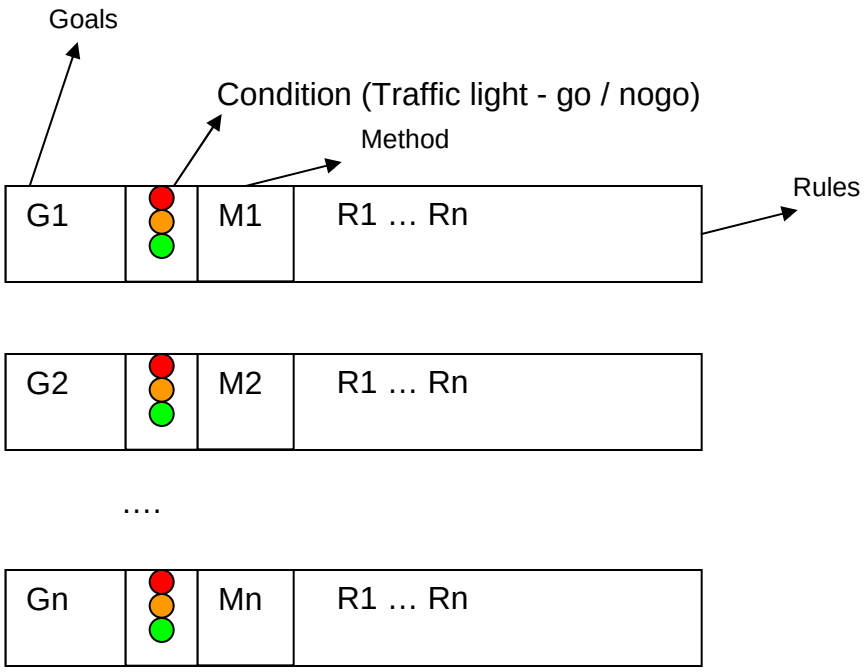


Figure 6 Goals, Conditions, Methods, Rules in a MAS (From [44])

An MAS can be designed and implemented using the following basic steps [44]:

1. Define the MAS Model.

A MAS can be constructed from the following model:

MAS (multiple agent systems) = { E (Environment), A (Agents), O (Objects), Ops (Operations), Laws }

i. E (Environment)

First, the environment the agents interact with is defined. This includes characteristics that provide sensory input for agents as well as characteristics affected by agents' actions.

ii. A (Agents)

Next, agents' specifications are described; this can include personality, activity preferences, and other attributes that will influence the agent decision making and action.

iii. O (Objects)

Objects are the other entities in the environment that agents interact with. The object attributes that affect the agents' sensory input and that are affected by agent actions are described. In this thesis, only agents with adaptable behavior are characterized as "agents" and afforded in-depth elaboration, while other sensing and acting entities are categorized and described as "objects."

iv. Ops (Operations)

Operations are actions that can be carried out by agents and objects. The sequences of cause and effects are described.

v. Laws

Laws are the boundary conditions that limit the agents' scope of operation. Laws can include physical space, resource, and operational limitations.

2. Define the Experiment

The experiment consists of the null hypothesis, the alternate hypothesis, and the measures to be collected for statistical analysis. The alternate hypothesis describes the objective of the experiment. The null hypothesis describes an opposite or different assumption that when refuted with statistical data can be used to prove the original hypothesis. [45]

2. Relevant Work

Harney's thesis is an early work describing an implementation of agent-based entities in a port-security scenario with X3D graphics [5]. Harney described entities that are able to move, change speed and direction, avoid collision, intercept, attack, and defend. Sullivan extended Harney's work to design and implement agent entities in the port using DES with Simkit [7]. Oliver Tan's thesis explored the use of a multi-agent system with cognitive blending for tracking the intentions of surface contacts in ports and waterways. Simkit was

used as the foundation for the discrete-event simulation of the port entities and the connector-based multi-agent simulation library (CMAS) was used for the agent implementation [6].

These efforts focused on port security against a non-adaptive agent for the small-boat threat. In this thesis, enhancing the adaptability of the small-boat threat is a logical step in the evaluation of port-security measures.

D. ADAPTABILITY IN BEHAVIOR

Hu demonstrated context-dependent adaptability in crowd-control behavior using multi-agents in a crowd-control simulation framework [46]. The agents make different decisions for choosing their behaviors depending on behavioral context. Switching to a different context (S) in the top behavioral-context layer modulates behavior choices in the lower behavior layer, as shown in the figure below. To make a behavior choice, Hu considers each behavior choice (b) in turn, taking the current sensory-input excitation of that choice and inhibiting it with the previous activation level of the other behavior choices. Inhibition relationships are predefined between the behaviors in the behavior layer, and there is a different set of inhibition relationships for behavioral context. Modulation achieves the behavioral-context switch by switching to the corresponding set of inhibition relationships.

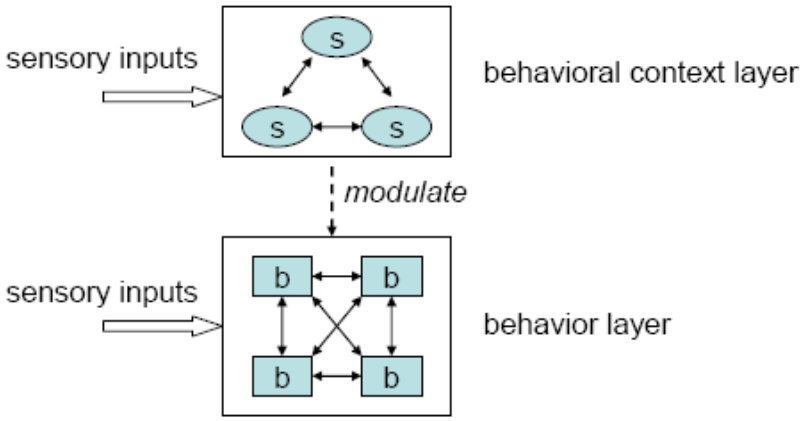


Figure 7 The Two-Layer Behavior Selection Architecture (From [46]).

This thesis applies similar concepts by using

i) a two-layer personality-action (behavioral context-behavior) concept,

ii) a switch to the corresponding action set for the personality, and

iii) modulation to modify the probability of an action based on the happiness of the previous same-action choice. The probability of other actions in the set are reduced proportionally. The happiness of the previous action choice depends on how happy the last chosen personality is with the results. This is computed from the last personality choice and current sensory inputs.

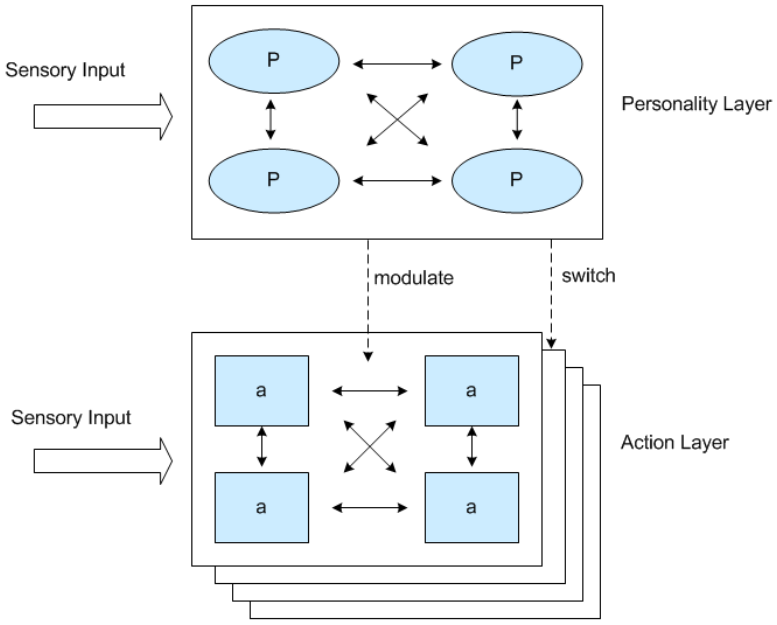


Figure 8 The Two-Layer Action Selection Architecture (After [46]).

E. DESIGN AND EXPERIMENTAL APPROACH

The adaptable MAS small-boat-threat behavior is designed using the MAS model and built upon the DES framework. The standard small-boat threat behavior uses the same DES framework and is constructed based on a flowchart of potential small-boat-threat tactical choices. In the next chapter, the design of the DES and MAS for the simulation is enlarged upon.

The experiment design is elaborated in Chapter IV and defines the null hypothesis that the MAS small-boat threat does not demonstrate adaptability, the alternate hypothesis that the MAS agent demonstrates adaptability, and the measurements for adaptability. The results are then collected and analyzed.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN OF THE MAS-DES PORT SECURITY

A. DES DESIGN

The DES design for the port-security simulation has four key categories: setup, movement, sensors, and engagement. These are described in the following sections.

1. Setup

The following diagram shows the setup-event graph depicting the connections between the arrival, creator, and manager components.

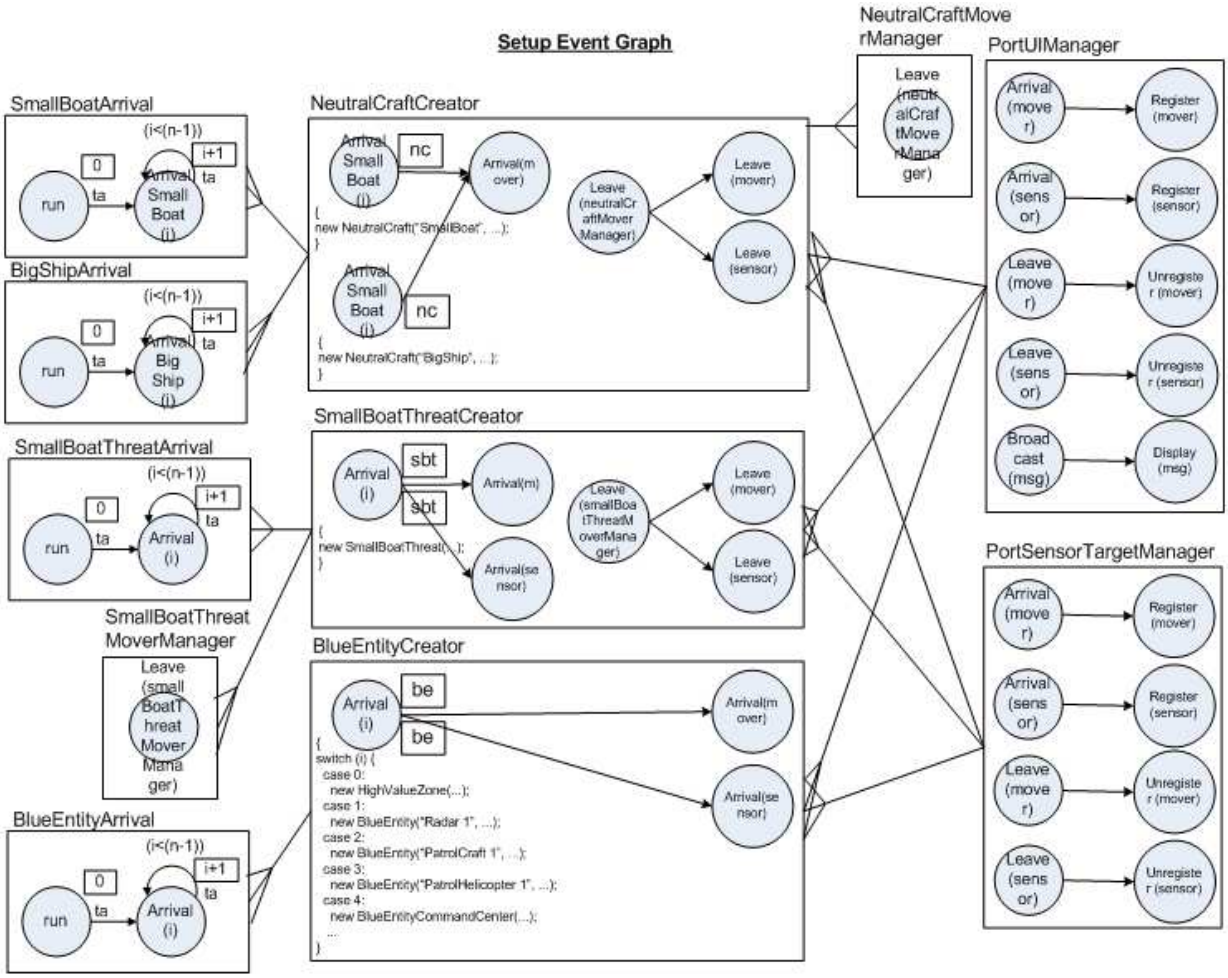


Figure 9 Setup-Event Graph

The creator components listen to the corresponding arrival components and creates the corresponding entities when the Arrival events are heard. The manager components listen to the creator components and register the entities for sensing and displaying when the Arrival events are heard. The creator components also listen to the Leave events of the mover managers and schedules the Leave event for the attached sensor and mover entities. The manager components will hear this Leave event and unregister the entities.

2. Movement

The following diagram shows the movement-event graph depicting the connections between the mover, mover manager, and engagement components.

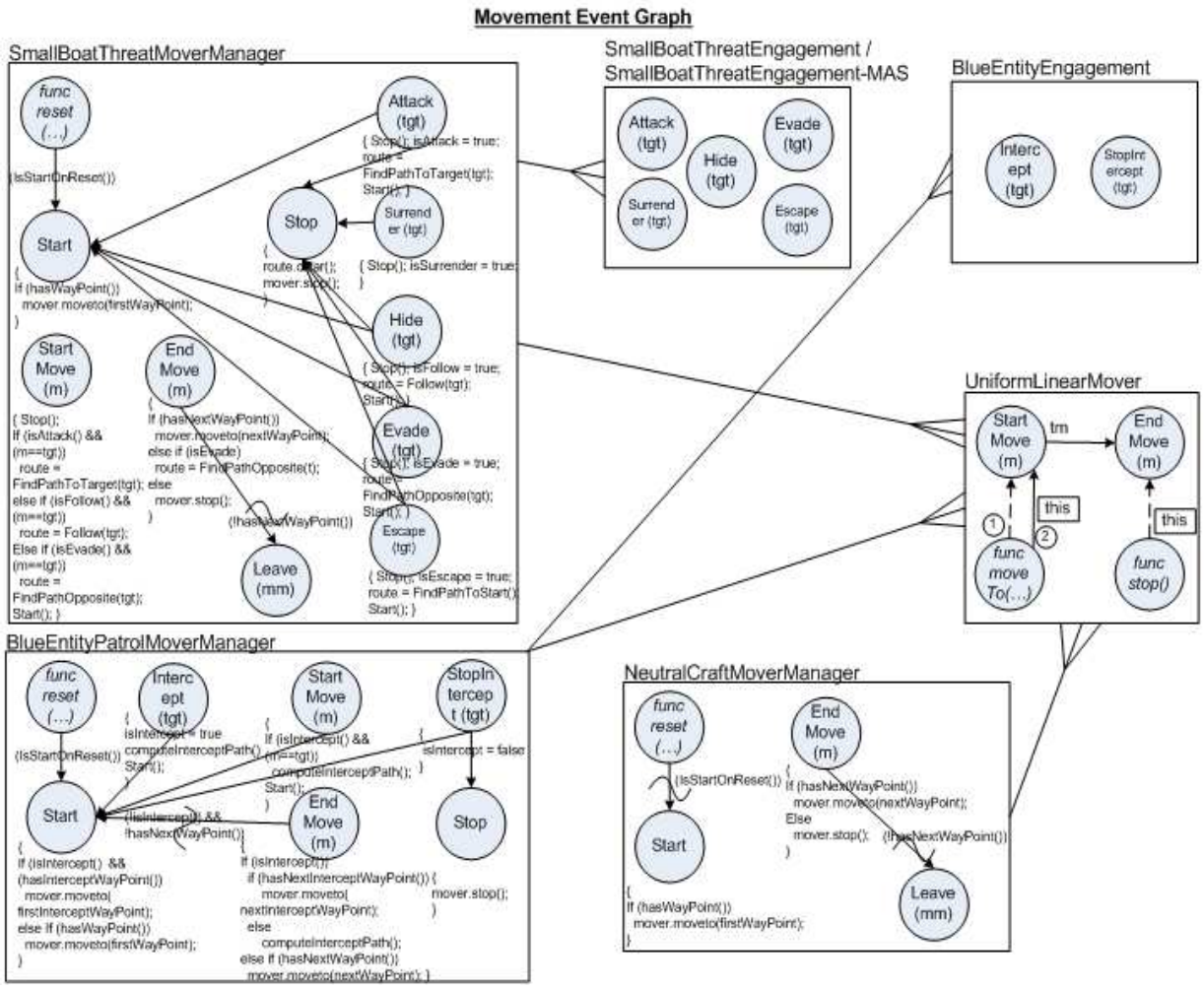


Figure 10 Movement-Event Graph

The mover-manager component directs the mover component to move to the next waypoint and listen to the mover component's EndMove event to know when the mover has arrived. The mover-manager components also listen to the movement-command events from the corresponding engagement components. For example, the small-boat-threat mover manager listens to the Attack, Hide, Evade, Escape and Surrender events from the small-boat-threat-engagement component. Similarly, the blue-entity-patrol mover manager (BEPMM) listens to the Intercept and StopIntercept event from the blue-entity-engagement component. The neutral-craft mover manager does not listen to any engagement component and simply directs its mover to go along the specified path.

The BEPMM extends the PatrolMoverManager and provides intercept behavior in addition to patrol behavior. When intercept is activated, the BEPMM stops patrolling behavior and runs intercept behavior. When intercept is called off, the BEPMM resumes patrol behavior.

The small-boat-threat mover manager (SBTMM) extends the PathMoverManager and uses the path-movement behavior to provide attack, following, evasion, escape, and surrender behavior. When the small-boat threat has reached the destination for attack or escape, or when it has surrendered, it will schedule the Leave event to initiate removing the threat from the simulation.

3. Sensor

The following diagram shows the blue-entity flowchart for small-boat-threat detection and engagement procedures. As can be seen, the detection process follows the detect→classify→recognize→identify cycle. The detected entity is also immediately tracked to monitor its speed and entrance into high-value zones.

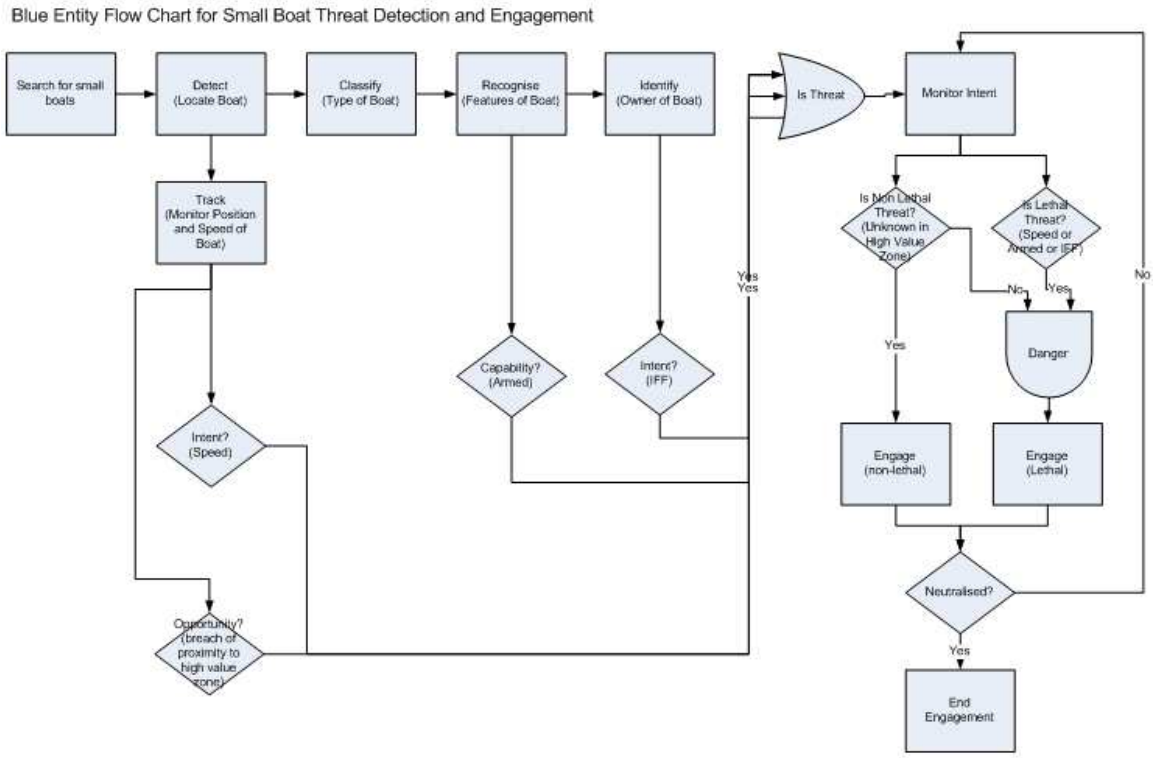


Figure 11 Blue-Entity Flowchart for Small-Boat-Threat Detection and Engagement

The following diagram shows the blue-entity sensor-event graph depicting the connections between the sensor manager, the sensor referees, the sensor mediator, the blue-entities sensors, the movers, and the blue-entity command center. In the setup-event graph, the sensor manager listens to the Arrival and Leave events and notifies the referees to register and unregister the specified entities. The sensor manager also registers the mediators in the Mediator framework for all combinations of sensors and movers in the simulation.

The referees listen to the movementState properties of the registered sensors and movers and computes the time delay for subsequent EnterRange and ExitRange events. When the mediator hears the EnterRange or ExitRange events, it checks whether the sensed entity is hidden by another entity, and schedules the Detection and Undetection events in the detecting sensor if not.

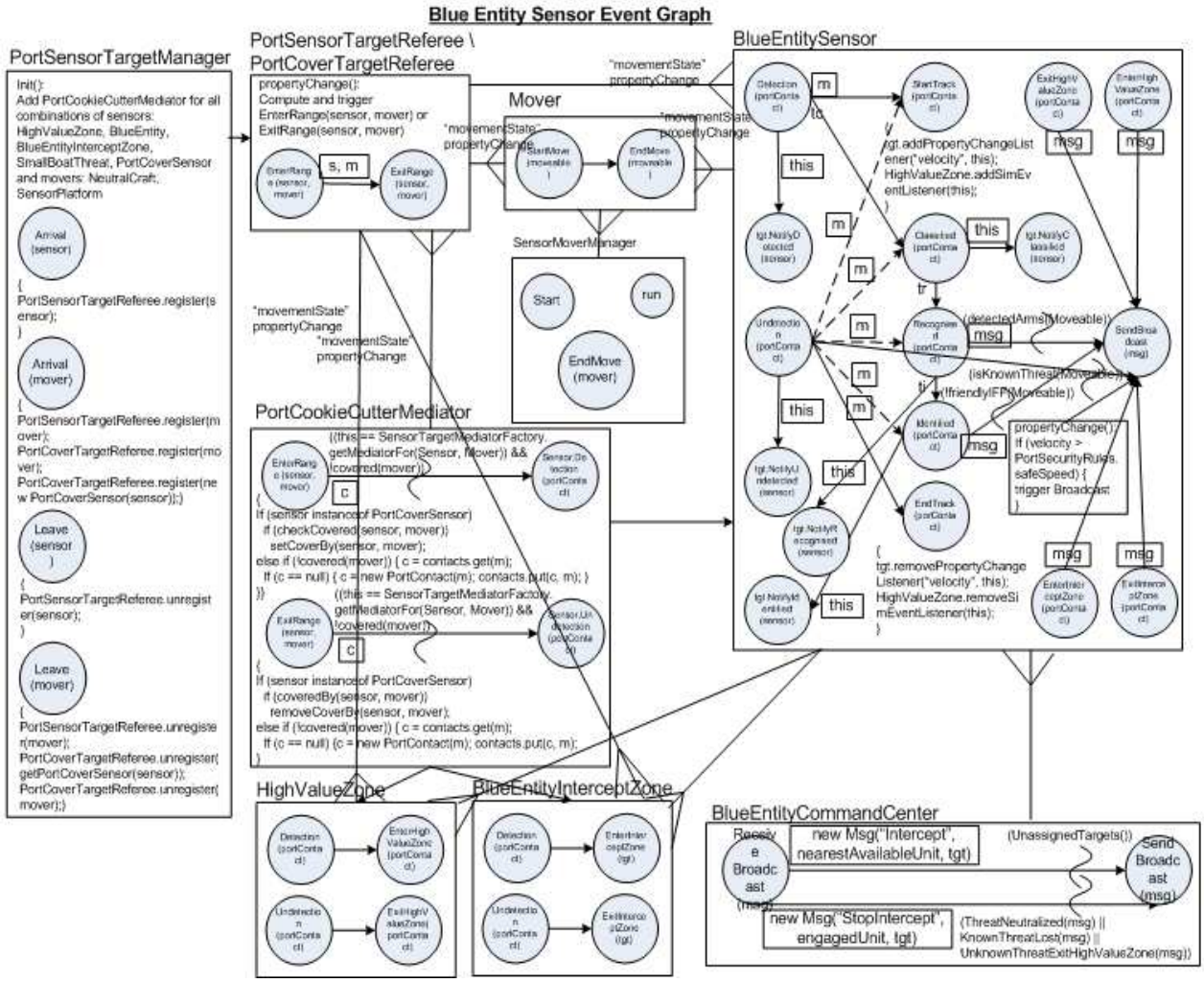


Figure 12 Blue-Entity Sensor Event Graph

The blue-entity sensor implements the detect→classify→ recognize→identify cycle. In addition, it tracks the velocity change of the target and whether it has entered a high-value zone. It does so by listening to the velocity property change of the target and by listening to EnterHighValueZone and ExitHighValueZone events of the high-value zone. The high-value zone is a stationary sensor and informs listening entities when a target enters its sensing radius.

The blue-entity intercept zone is a mobile sensor that attaches itself to the same mover platform as the blue-entity sensor. The blue-entity sensor listens to the blue-entity intercept zone to know when a target has entered its intercept

radius, which is different from its sensor radius. Finally, the blue-entity sensor sends SendBroadcast events to the listening blue-entity command center to inform it of these detections, and the command center will assign target interception to whichever blue entity can get there fastest.

The following diagram shows the small-boat-threat flowchart for blue-entity threat detection and engagement procedures. Like the blue entity, the small-boat threat also follows the detect→classify→recognize→identify cycle, but is able to identify the patrol on completion of the recognized phase. The small-boat threat is also able to track the speed of the target to identify the blue entity.

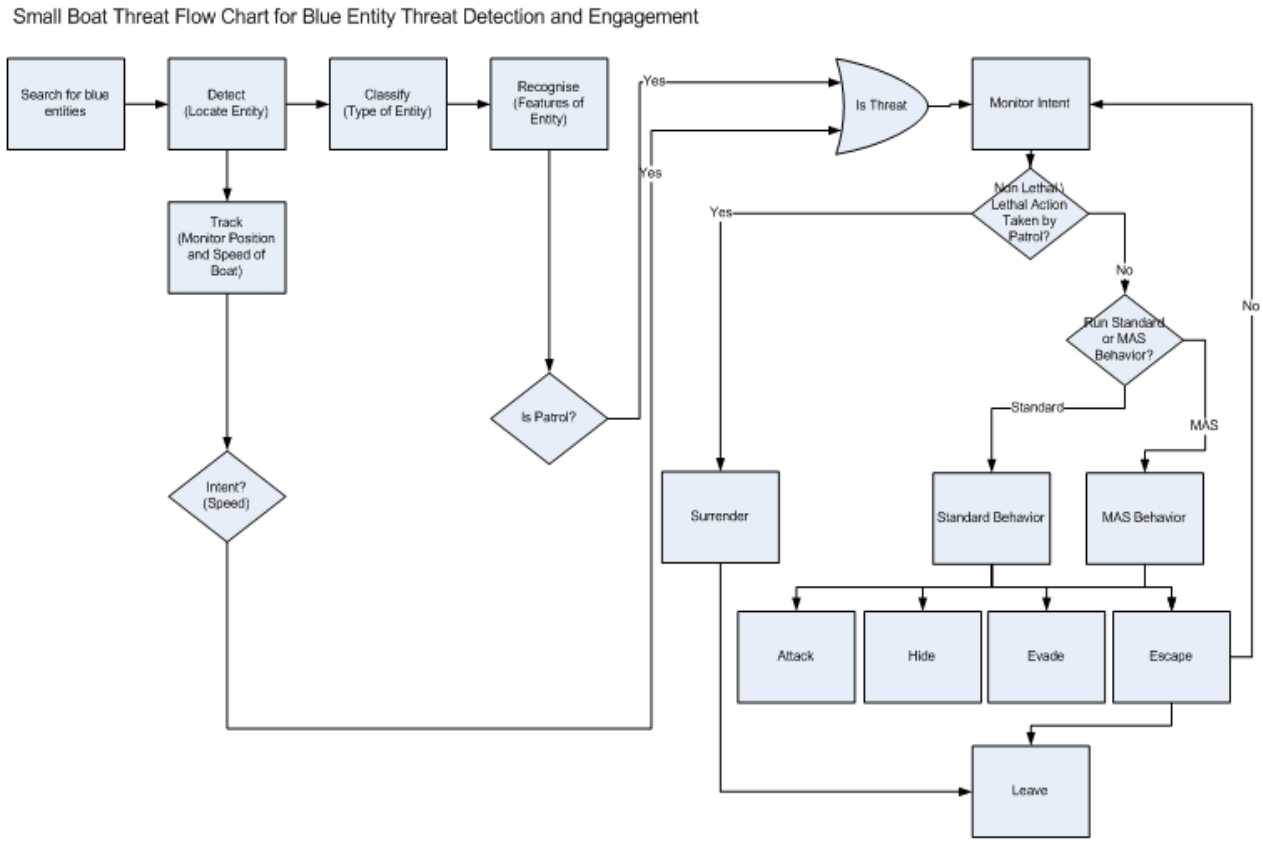


Figure 13 Small-Boat Threat Flowchart for Blue-Entity Detection and Engagement

The following diagram shows the small-boat-threat sensor-event graph depicting the connections between the sensor referees, the sensor mediator, the small-boat-threat sensor, the movers, and the port-cover sensor.

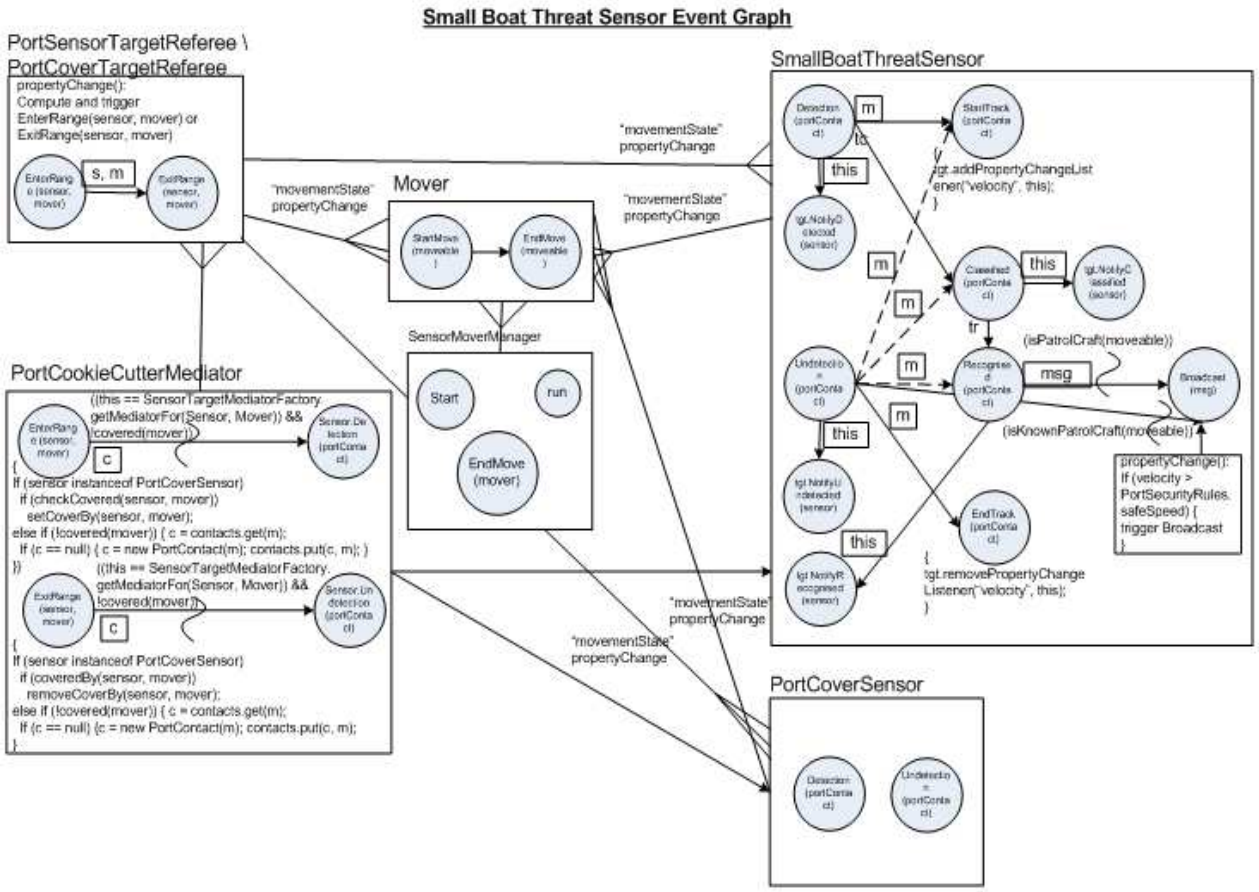


Figure 14 Small-Boat-Threat Sensor-Event Graph

The following diagram shows the flowchart for the integrated sensor and cover-detection process. Each detectable entity can enter into four states:

i) within sensor range and cover range,

ii) within sensor range and outside cover range,

iii) outside sensor range and within cover range, and

iv) outside sensor range and cover range.

In the first state, the entity is visible to the sensor, while the in the other three states, the entity is hidden.

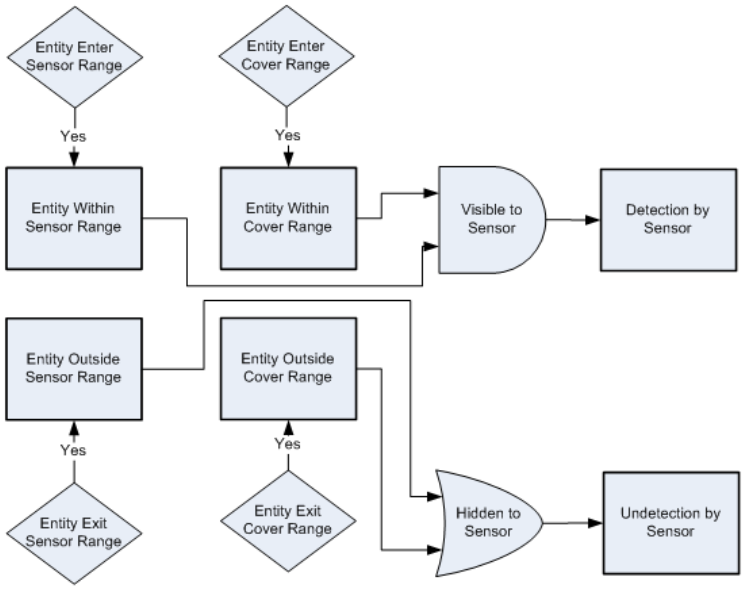


Figure 15 Flowchart for the Integrated Sensor and Cover-Detection Process

The following diagram shows a simplified view of the cover mediator working jointly with the normal sensor and cover sensor to decide whether to inform the sensor that it has detected an entity. Refer to Appendix B: Code Snippets for sample implementation code.

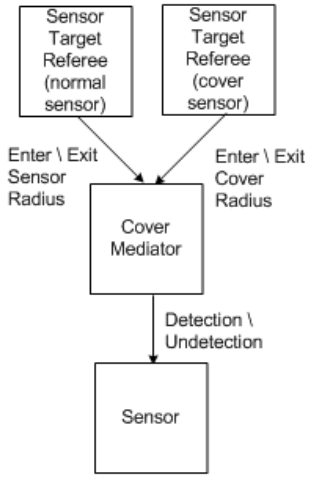


Figure 16 Simplified View of Cover Design with Cover Mediator and Cover Referee

When the mediator hears the EnterRange event, it checks whether the event is to signal one entity covering the other; it does this by checking whether

the sensor is a cover sensor, and if so, checks their relative radar cross section to decide which entity provides the cover. If both have the same relative radar cross section, the current sensor is chosen to provide the cover. When the mediator hears the ExitRange event for a cover sensor, it checks and removes any cover that is provided between the entities.

The following is a sample scenario of cover provided by an entity with a larger relative radar cross section.

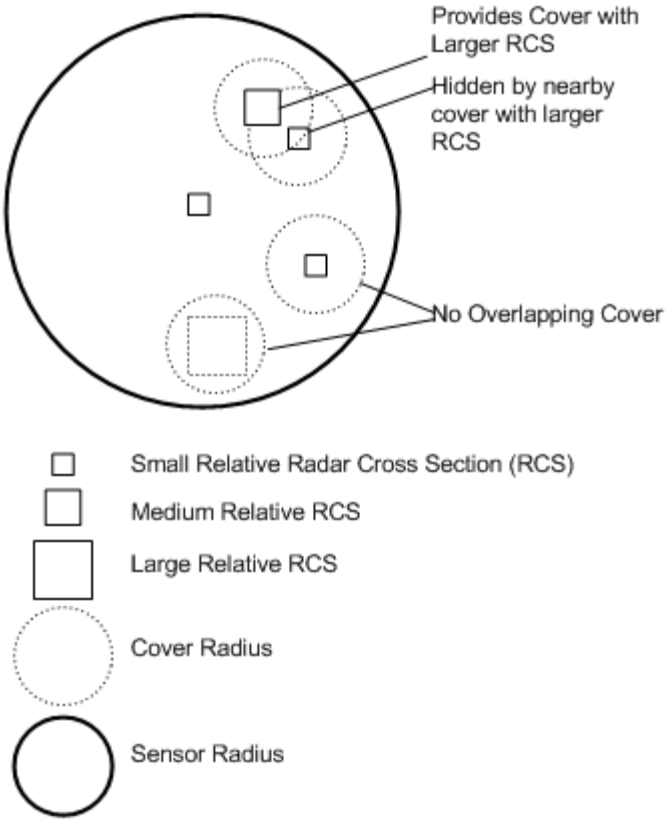


Figure 17 Providing Cover with Relative Radar Cross Sections

The small-boat-threat sensor sends Broadcast events of these detections to its engagement component, and this provides the necessary sensory input to the engagement component.

4. Engagement

As shown in Figure 11, the blue entity will continue to move towards the small-boat threat to apply nonlethal or lethal force until the threat is neutralized. As shown in Figure 13, the threat will surrender when the blue entity applies force, whether lethal or nonlethal. Otherwise, the threat will run the standard or MAS behavior to choose between attacking, hiding, evading, and escaping.

The following diagram shows the engagement-event graph depicting the connections between the mover-manager, engagement, intercept-zone, and command-center components.

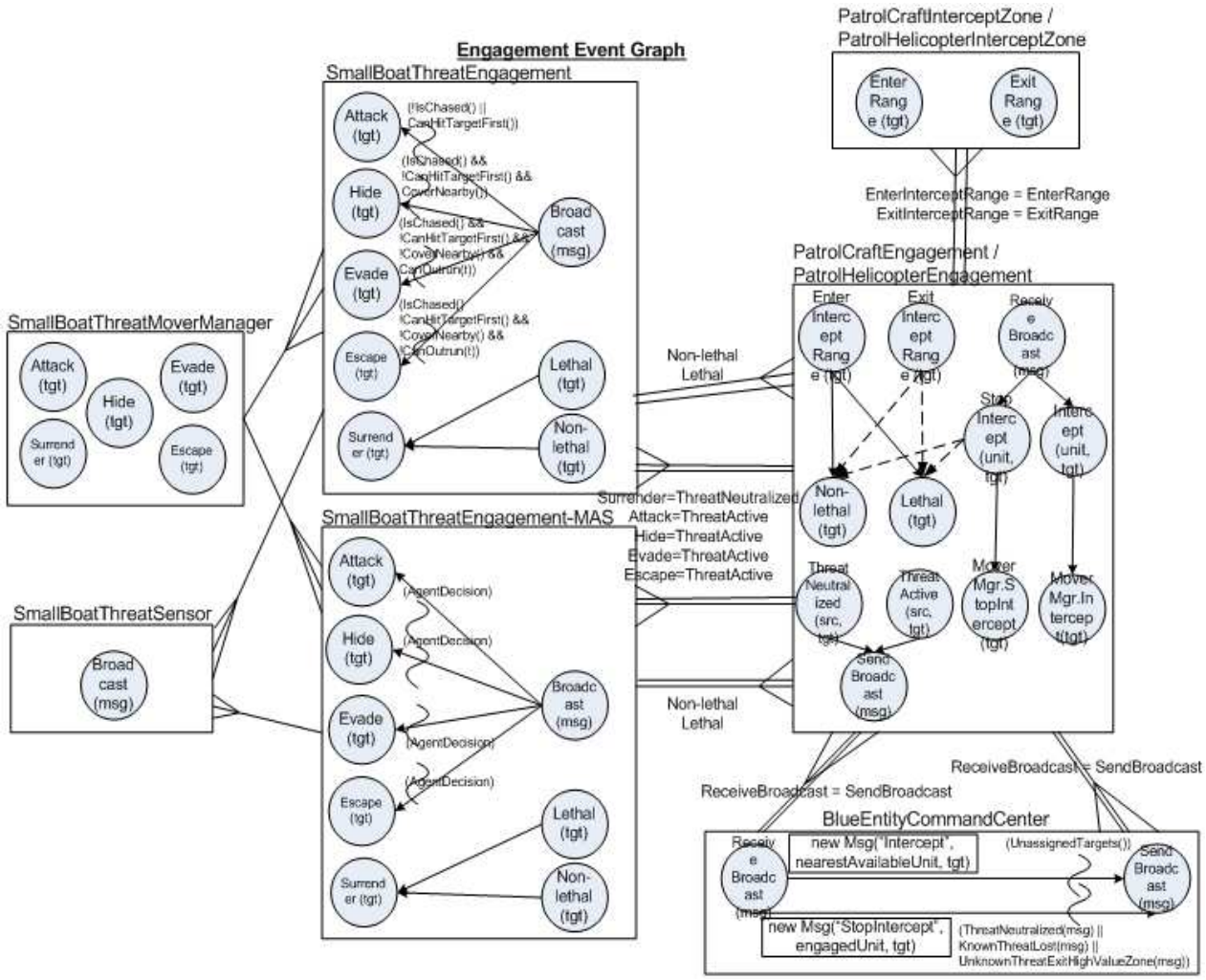


Figure 18 Engagement-Event Graph

The patrol-engagement component listens to command broadcasts from the blue-entity command center to intercept or stop intercept of a target. It notifies its mover-manager component to carry out these actions. When the target has entered its intercept zone, the patrol-engagement component hears the EnterInterceptRange event and takes non-lethal or lethal action on the target. The small-boat threat engagement component hears these Non-lethal and Lethal events and immediately schedules the Surrender event. The patrol-engagement component listens for the small-boat threat's actions and informs the command center whether the small-boat threat is neutralized or active.

The small-boat-threat-engagement component listens for sensor broadcast events from its sensor component and runs the standard or MAS behavior to select and schedule the appropriate action event, including Attack, Hide, Evade or Escape. This is picked up by the listening mover-manager component, which carries out the corresponding movements.

B. MAS DESIGN

The small-boat threat MAS model is defined as

Small-boat threat MAS Model =

{

E (Port of Oakland),

A (Small-boat threats),

O (Patrol crafts and helicopters, radar, big ships, small boats, high-value zones),

Ops (Sensing, navigation, small-boat threat activities, blue-entity activities, results of activities),

Laws (Two-dimensional world, sensor laws, movement laws, activities laws)

}.

The following diagram shows an overview of the small-boat threat MAS model.

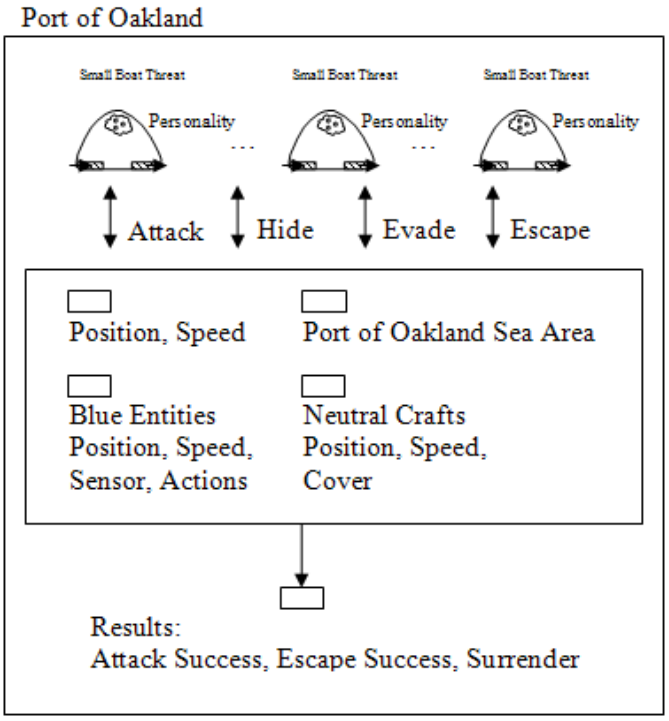


Figure 19 Overview of Small-Boat Threat MAS

The follow sections describe the environment, agents, objects, operations and laws of the small-boat threat MAS model in more detail.

1. Environment

a. Port of Oakland

The Port of Oakland environment in “Port Security 2012” [1] was used as the starting point for this thesis. The sea area in the Port of Oakland defines the sea entities’ area of movement. A pathfinding map covering this area is required for mobile agents, especially the patrol helicopter, patrol crafts, and small-boat threats. As “Port Security 2012” uses only fixed paths, there is a need to generate this pathfinding map in this thesis.

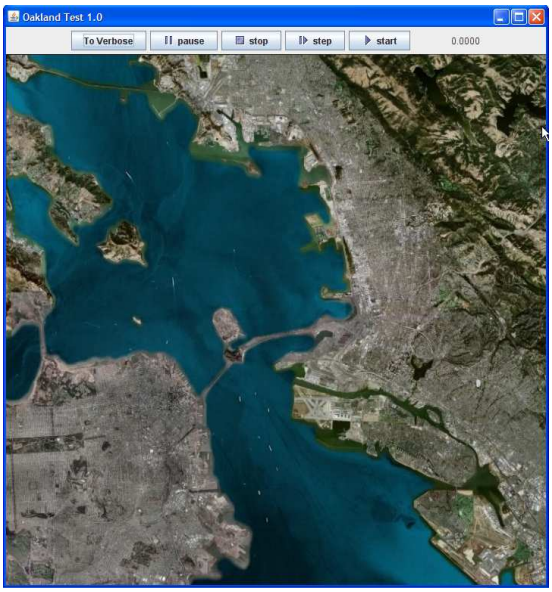


Figure 20 Oakland Satellite-Imagery Map from Google Map

b. Automated Generation of Pathfinding Map of Port of Oakland

An automatic map-generation process for creating an A* pathfinding map was designed and implemented in this project. This A* pathfinding map defines sea locations that vessels can navigate to and enables the A* search to find the path. Manually defining this pathfinding map would be tedious; instead, automatic map generation computes the map from shoreline data downloaded from the National Oceanic and Atmospheric Administration (NOAA) public website [38]. The result is an eight-connected grid map of the Port of Oakland. Snapshots of the process are shown in the figures below.

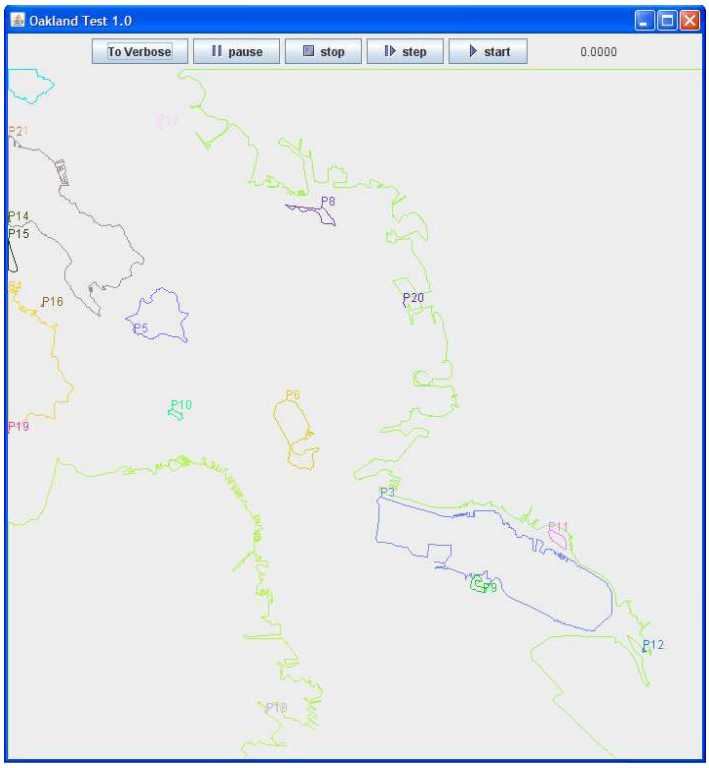


Figure 21 NOAA Oakland Shoreline Vector Map

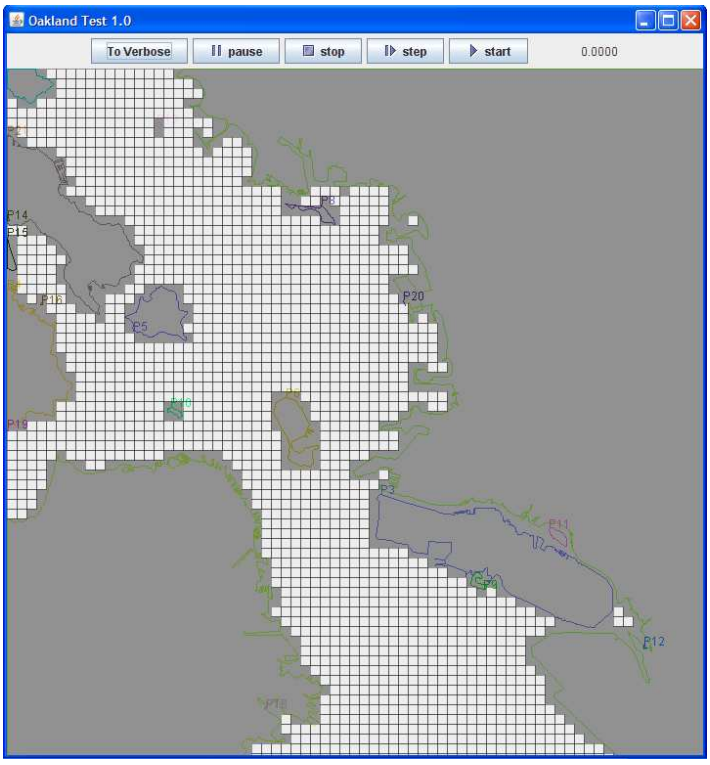


Figure 22 Grid Cell Map Generated from Shoreline Vector Map (Shown with Shoreline Vector Map)

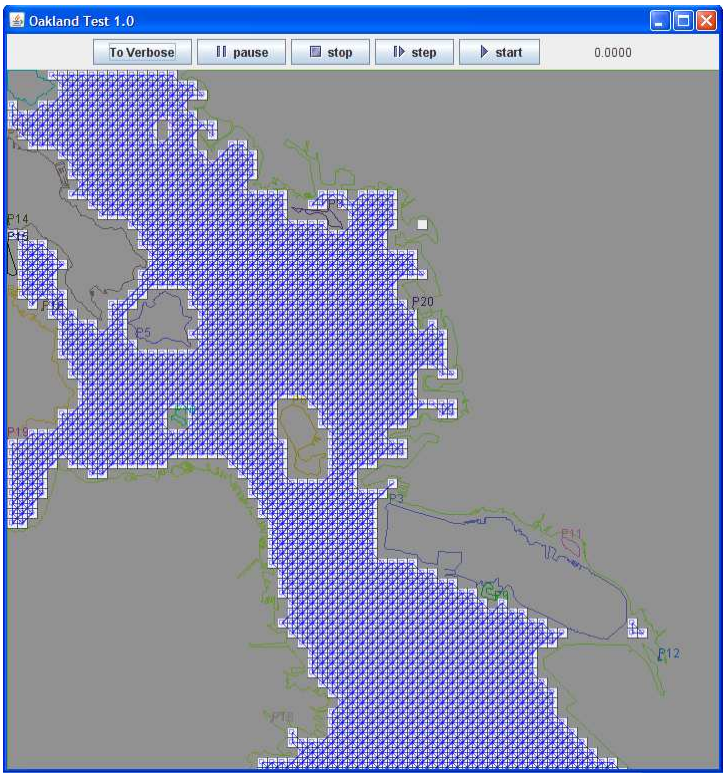


Figure 23 Eight-Connected Graph Generated from Grid Cell Map (Shown with Grid Cell Map and Shoreline Vector Map)

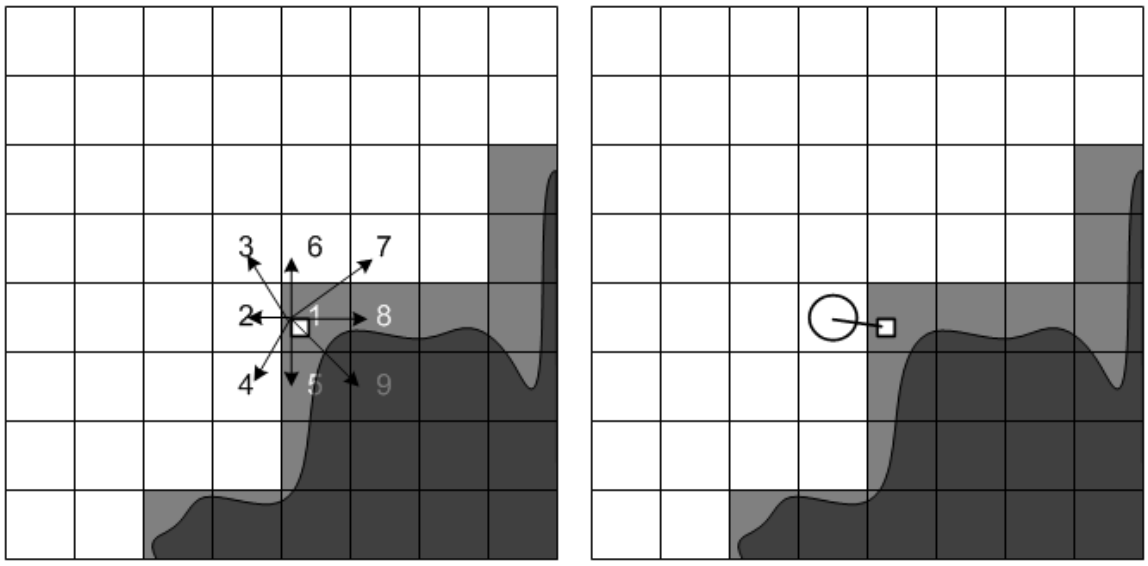
A grid-cell map representation as described by Tozour [41] is first generated from the Oakland shoreline vector map. An eight-connected graph is then generated from the grid-cell map. The eight-connected graph is chosen over the four-connected graph because it improves A* performance with the more direct route available to the destination. The automated search space representation consists of the following key steps:

1) Separation of land and sea mass through creation of land mass polygons using NOAA shoreline vector map.
2) Map scaling and positioning of Port of Oakland satellite raster map and shoreline vector maps using simple map processing.
3) Computation of grid map through land-mass check of shoreline vector map. Land mass is defined as obstacle grid cells in the grid map.
4) Computation of network graph through creation of nodes and edges based on non-obstacle grid cells in the grid map.

c. Dynamic Movement of Sea Entities in the Port of Oakland

Sea entities are able to move dynamically and without collision into the land mass in the Port of Oakland by using the following algorithm:

(1) Finding Nearest Waypoints to Starting and Ending Points. The nearest grid cells in the pathfinding map to the starting and ending points are first located. An algorithm is designed and implemented to take the input points to search the current grid cells and neighboring grid cells for non-obstacle grid cells. This finds the non-obstacle starting and ending grid cells that is required for the A* pathfinding search to be able to find a valid path. This is shown in the diagram below. The algorithm searches the current grid cell, followed by the nearest left, right, bottom or top grid cells, followed by the diagonal grid cells, and then, finally, clockwise from the bottom cell.



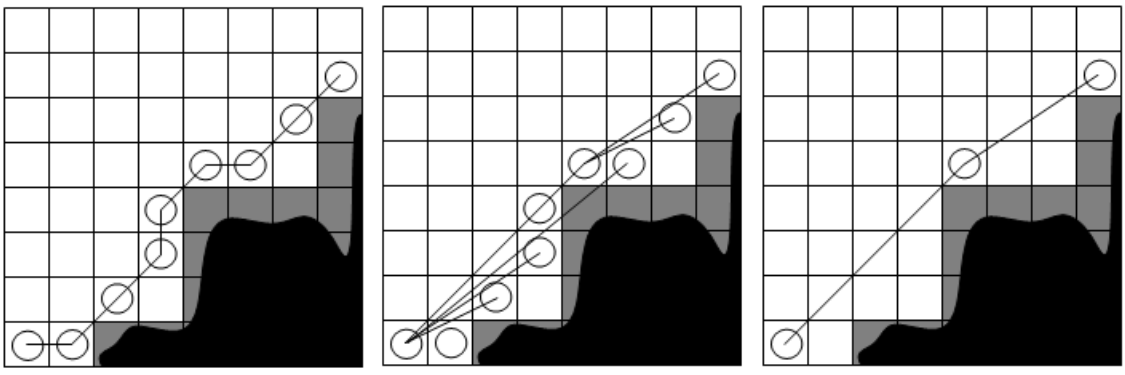
(a) (b)

Figure 24 Finding Nearest Waypoint to Starting and Ending Points: (a) Grid cell search order for nearest waypoint; (b) Path to nearest waypoint

(2) Finding a Near-Optimal Path with A* Pathfinding Algorithm. The A* pathfinding algorithm in Simkit\Diskit is applied to the generated pathfinding map and tested in this thesis. This computes a path through the centers of the non-obstacle grid cells in the pathfinding map.

(3) Connection of Starting and Ending Points. The starting and ending points are appended to the ends of the A* path, to complete the path from the starting point, through the pathfinding map, and to the ending point.

(4) Path Smoothing. The resulting path consists of a large number of points, many of them unnecessary in the Port of Oakland scenario, which consists of large sea masses. Path smoothing is performed as a post-processing step to remove unnecessary points. A straightforward path smoothing described by Pinter is implemented and modified [42]. Pinter's algorithm checks whether intermediate waypoints can be removed without having the path crossing the blocked grid cell. This is shown in the diagram below. The black polygon represents the shoreline obstacle and the light gray squares are the corresponding blocked grid cells. Note that Pinter's algorithm reduces the number of waypoints returned by the A* pathfinding algorithm while avoiding the blocked grid cells.



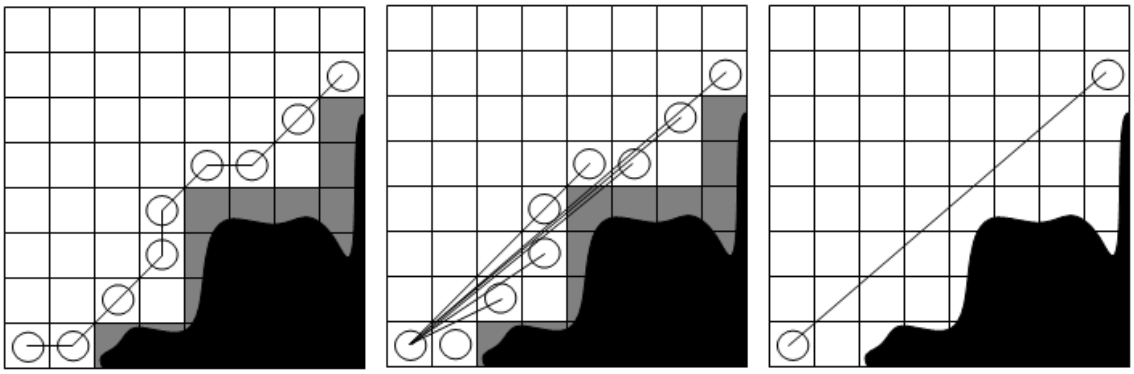
(a) (b) (c)

Figure 25 Pinter's Path Smoothing: (a) Typical A* path with many waypoints; (b) Check for smoothing against blocked grid cells (gray); (c) Final path with intermediate waypoints removed

The pseudo-code for Pinter's algorithm is shown below.

```
checkPoint = starting point of path
currentPoint = next point in path
while (currentPoint→next != NULL)
    if CanSmoothSegment(checkPoint, currentPoint→next)
        // Remove intermediate points to
        // make a straight path between those points:
        temp = currentPoint
        currentPoint = currentPoint→next
        delete temp from the path
    else
        checkPoint = currentPoint
        currentPoint = currentPoint→next
```

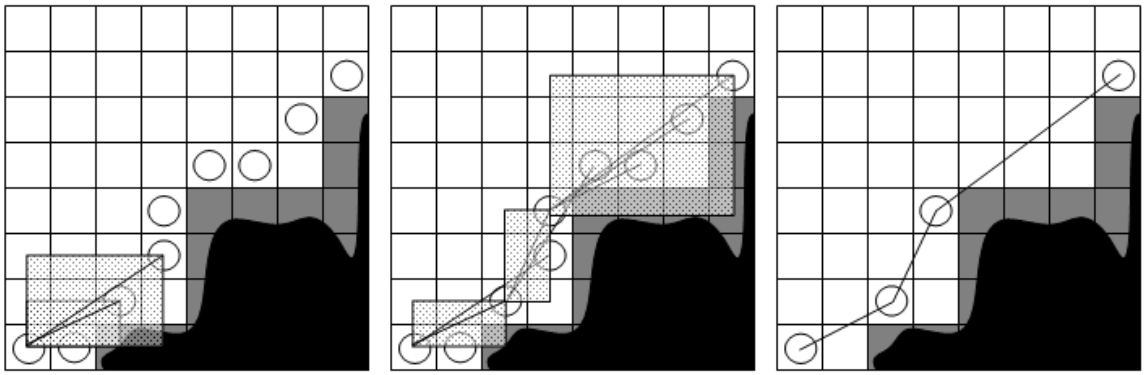
In this thesis, the "CanSmoothSegment" function is modified from Pinter's algorithm to consider the finer-resolution shoreline obstacle. This is shown in the diagram below. This precise algorithm reduces the number of waypoints returned by the A* pathfinding algorithm while avoiding the shoreline, but the computation takes a long time, because the checks for each line segment against the shoreline polygon are computer-intensive.



(a) (b) (c)

Figure 26 Precise Smoothing: (a) Typical A* path with many waypoints; (b) Checks for smoothing against shoreline (black); (c) Final path with intermediate waypoints removed

A basic obstacle-checking algorithm is chosen over a more precise one for better performance in the path-smoothing algorithm. This basic obstacle-checking algorithm inspects shoreline intersections with segment bounds, allowing smoothing against the shoreline while maintaining acceptable performance.



(a) (b) (c)

Figure 27 Basic Smoothing: (a) Checks of segment bounds against shoreline; (b) Bounds of non-intersecting segments; (c) Final path with intermediate waypoints removed

2. Agents

a. Small-Boat Threat Attributes

The small-boat threat has four navigation modes, or actions, that the small-boat threat can take in response to sensory inputs: attack, hide, evade, and escape. Two small-boat threats start their attack during the first attack wave, and two others start during the second wave, five minutes later. The threat travels at full speed during the attack and can reduce speed for hiding and following its cover target. The small-boat threat is assigned a relatively low radar cross section, as this is identified as a potential detection-reducing tactic.

Table 1 Small-Boat Threat Attributes

		Navigation	Sensing Range (km)	Engagement Range (km)	Movement (knots)	Radar Cross Section (relative units)
Red Entities	4 Small-boat threats	●Initial - 2 small-boat threats start attack in first wave, 2 small boats start attack in second wave 5 minutes later ●Attack - move at maximum speed to attack target with dynamic pathfinding ●Hide - move at maximum speed to nearest cover and follow cover movement and speed with dynamic pathfinding ●Evade - move at maximum speed in general opposite direction from approaching blue entity with dynamic pathfinding ●Escape - move at maximum speed to starting location with dynamic pathfinding	1	Nil	●45 (max) ●0 to 45 (varies)	1

b. Small-Boat Threat Personality

Four personalities are potentially in play: the suicidal, tactical, deceptive, and balanced. The personalities accept sensory inputs to the personality layer to compute the happiness of the last decision and predict the happiness of the next. These happiness values are used to modulate the action weights and switch to the personality action weights for the next decision, as shown in the diagram below.

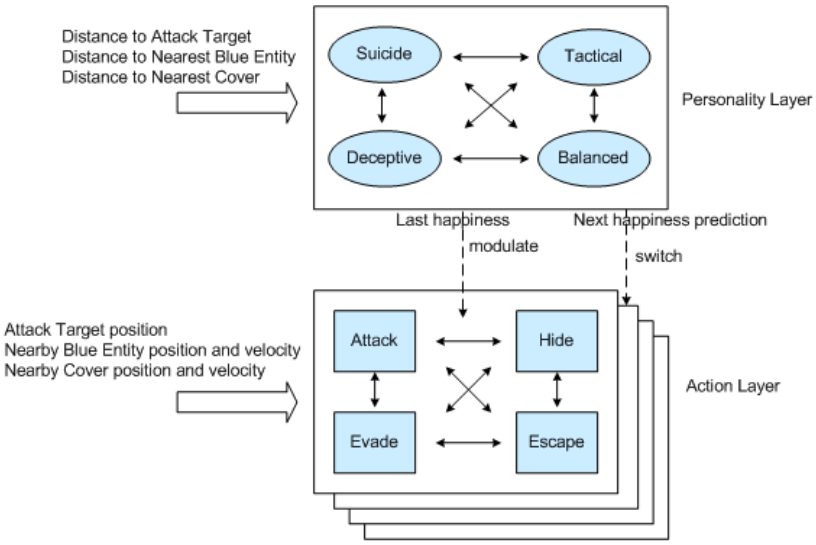


Figure 28 Personality and Action Selection

The sensory inputs to the personality layer are distance to the attack target, nearest blue entity, and nearest cover. Happiness is a function of the changes in these distances and their corresponding personality motivation weights. Happiness is computed as follows:

Happiness =

*(+ve normalized reduction in distance to attack target) * motivation for attack +*

*(+ve normalized increase in distance to nearest blue entity) * motivation for tactical +*

*(+ve normalized reduction in distance to nearest cover) * motivation for deception.*

Normalization of the distances uses the maximum distance for each case, for example, the maximum distance to the target is from the starting location, and the maximum distance to the nearest blue entity and nearest cover is the maximum sensor range of the small-boat threat.

The following table lists the small-boat threat personality matrix. The values remain fixed throughout the simulation.

Table 2 Small-Boat Threat Personality Matrix

Personality	Motivation for Attack (reduce distance to attack target)	Motivation for Tactical (increase distance from nearest blue)	Motivation for Deception (reduce distance to nearest cover)
Suicide	1.0	0.0	0.0
Tactical	0.0	1.0	0.0
Deceptive	0.0	0.0	1.0
Balanced	0.3	0.3	0.3

The following table lists the initial small-boat-threat action matrix. The weights in the action matrix are modified by the last happiness result. A positive happiness for an action increases its weightage and increases the probability that this action will be selected in subsequent decisions.

Table 3 Initial Small-Boat Threat Action Matrix

Personality	Attack Action Weights	Hide Action Weights	Evade Action Weights	Escape Action Weights
Suicide	0.25	0.25	0.25	0.25
Tactical	0.25	0.25	0.25	0.25
Deceptive	0.25	0.25	0.25	0.25
Balanced	0.25	0.25	0.25	0.25

The action matrix is updated using the following steps:

i) Half the value of the last happiness is added to the last chosen action of the last chosen personality.

ii) The remaining half of the happiness is divided equally between the remaining actions of this last chosen personality.

iii) The action weights of this last chosen personality are normalized so that they sum to one.

The pseudo-code for updating the action matrix is as follows:

*Action[last personality][last action] = (Action[last personality][last action] + 0.5 * happiness) / (1.0 + happiness)*

*Action[last personality][other actions] = (Action[last personality][last action] + 0.5 * happiness / 3) / (1.0 + happiness)*

There is no update for Action[other personalities][any action].

The action distribution for choosing the next action are computed thus:

Action Distribution { attack, hide, evade, escape } =

{

< Action[next chosen personality][attack],

< Action[next chosen personality][attack] + Action[next chosen personality][hide],

< Action[next chosen personality][attack] + Action[next chosen personality][hide] + Action[next chosen personality][evade],

< 1.0

}

The next action is chosen by selecting the action where the random-variate value falls under the action-weight distribution. This is subject to the rule conditions affected by the sensory inputs. Refer to Appendix B: Code Snippets, for implementation code.

c. Small-Boat Threat Goals, Conditions, Methods, Rules

The sensory inputs to the action layer are attack-target position, nearby blue-entity position and velocity, and nearby cover position and velocity. These inputs determine whether an action can be carried out, and how it is carried out physically in the Port of Oakland environment. The following table lists the small-boat threat goals, conditions, methods, and rules.

Table 4 Small-Boat Threat Goals, Conditions, Methods, and Rules

<table>
<tr><th>Goals</th><th>Conditions</th><th>Methods</th><th colspan="2">Rules</th></tr>
<tr><td></td><td></td><td></td><td>Conditions</td><td>Actions</td></tr>
<tr><td>Attack Target</td><td rowspan="3">1. Choose next personality based on greatest happiness for current situation (environment input)
2. Randomly chose action using chosen personality's action weight distribution</td><td rowspan="3">1. Compute happiness for last chosen personality
- Happiness = (+ve normalized reduction in distance to attack target) * motivation for attack +
(+ve normalized increase in distance to nearest blue entity) * motivation for tactical + (+ve normalized reduction in distance to nearest cover) * motivation for deception
- normalization based on maximum distance from start location to target, or maximum sensor range
2. Update and normalize last chosen action weight for last chosen personality based on happiness
- Add half happiness to last chosen action weight of last chosen personality
- Distribute remaining happiness to other action weights of last chosen personality
- Normalize action weights of personality</td><td>-</td><td rowspan="3">Attack or Hide or Evade or Escape</td></tr>
<tr><td>Avoid Being Intercepted</td><td>if (blue entity not within range), cannot choose evade or escape</td></tr>
<tr><td>Find Cover</td><td>if (cover not within range), cannot choose cover</td></tr>
</table>

To decide which action is carried out, the personality with the most happiness is chosen first. The action weights for decision making are switched to this personality's action set. A random variate is generated to select the action in this action set. If the action selected by the random variate does not satisfy the rule conditions, the next action is chosen and its rule-conditions are checked. This continues until a valid action is found. If no valid action is found, the default action is chosen. This default action is attack.

A sample decision scenario is presented below.

In the initial condition,

i) normalizing distance to attack target = distance from starting point to attack target = assuming 100km,

ii) normalizing distance to nearby blue entity = maximum sensor range = 1km,

iii) normalizing distance to nearby cover = maximum sensor range = 1km,

Assuming that in the current decision event,

i) reduction in distance from attack target is 10km,

ii) increase in distance to nearest blue-entity location is 0.05km,

iii) reduction in distance to nearest cover is 0.01km,

iv) last chosen personality is suicide, and

v) last chosen action is attack,

vi) last updated action weights values for suicide personality are [0.26, 0.249, 0.246, 0.245].

*Happiness for suicide personality = ($10km / 100km) * 1.0 + (0.05km / 1km) * 0.0 + (0.01km / 1km) * 0.0 = 0.1*

*Happiness for tactical personality = ($10km / 100km) * 0.0 + (0.05km / 1km) * 1.0 + (0.01km / 1km) * 0.0 = 0.05*

*Happiness for deceptive personality = ($10km / 100km) * 0.0 + (0.05km / 1km) * 0.0 + (0.01km / 1km) * 1.0 = 0.01*

*Happiness for balanced personality = ($10km / 100km) * 0.3 + (0.05km / 1km) * 0.3 + (0.01km / 1km) * 0.3 = 0.03 + 0.015 + 0.003 = 0.0453*

Thus, happiness for last personality (suicide) is 0.1. Chosen personality is also suicide as it has the largest happiness value.

The action matrix is updated as follows:

*Action[suicide][attack] = (0.26 + 0.5 * 0.1) / (1 + 0.1) ~= 0.282*

*Action[suicide][hide] = (0.249 + 0.5 * 0.1 / 3) / (1 + 0.1) ~= 0.242*

*Action[suicide][evade] = (0.246 + 0.5 * 0.1 / 3) / (1 + 0.1) ~=0.239*

*Action[suicide][escape] = (0.245 + 0.5 * 0.1 / 3) / (1 + 0.1) ~= 0.298*

The other values in the action matrix are unchanged.

This gives the following action distribution for the next chosen personality (suicide): Action Distribution { <0.282, < (0.282 + 0.242), < (0.282 + 0.242 + 0.239), <1.0} = Action Distribution { <0.282, <0.524, <0.763, <1.0 }

Assuming the random variate for the action choice generates a value of 0.123, this corresponds to choosing the attack action for the next behavior. Alternatively, assuming the random variate for the action choice generates a value of 0.345, this corresponds to choosing the hide action for the next behavior. This invokes the rule condition, which checks whether there is cover nearby. If there is cover nearby, hiding is chosen. Otherwise, the next action (evade) is considered and this also invokes the rule condition that there must be a blue entity nearby to evade. It there is a blue entity nearby, evasion is chosen. If not, the next action (escape) is considered and again, this invokes the rule condition that there must be a blue entity nearby to evade. It there is blue entity nearby, escape is chosen. If not, the default action (attack) is chosen.

d. Small-Boat Threat MAS Behavior

The following flowchart shows how the small-boat threat MAS behavior performs a personality and action selection, after which the selected action is carried out, subject to constraints of whether a blue entity or cover is nearby.

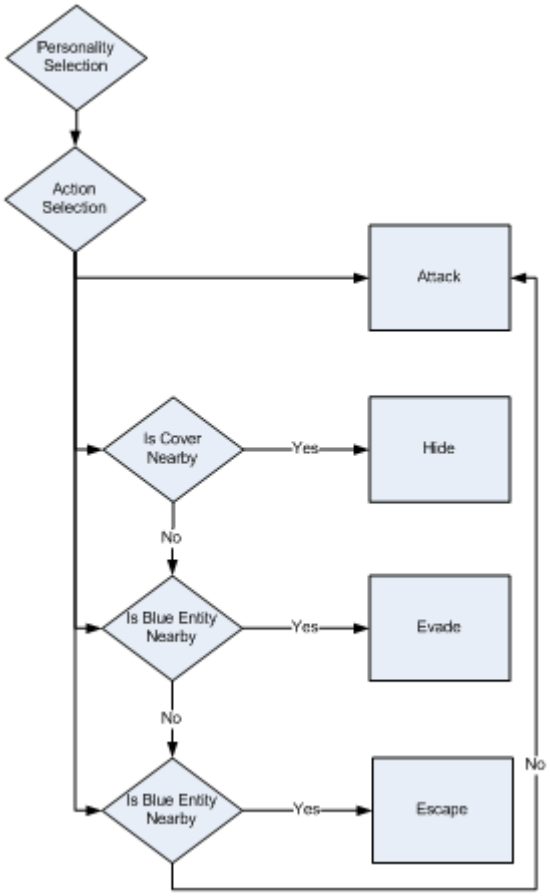


Figure 29 Small-Boat Threat: MAS Behavior Flowchart

e. Small-Boat Threat Standard Behavior

The same environment, objects, operations and laws are used by the small-boat threat standard behavior, and only the behavior is different from the threat MAS behavior. The flowchart below is the design for the standard behavior of the small-boat threat and is implemented using DES. It uses a few more rules, such as whether it is chased, can hit the target first, or can outrun the target, and the decision choice is made using this fixed sequence of rules.

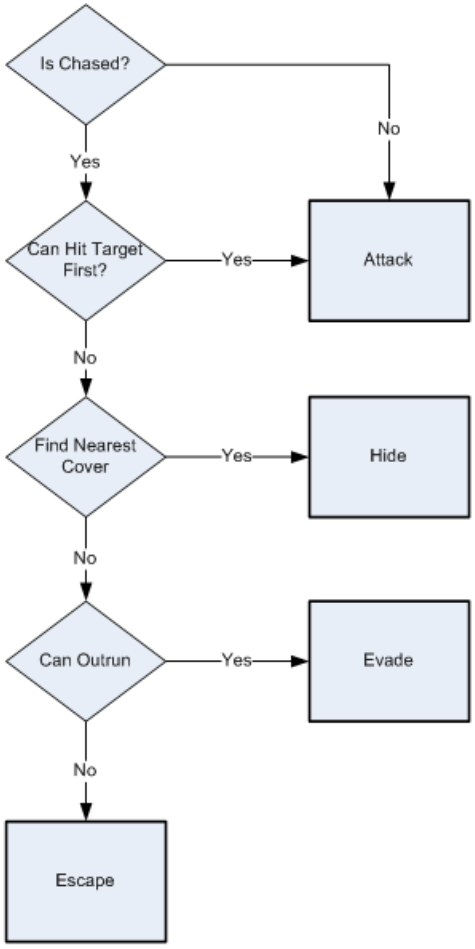


Figure 30 Small-Boat Threat: Standard Behavior Flowchart

3. Objects

The blue entities consist of one patrol helicopter, four patrol crafts, one stationary radar, and three high-value zones. The neutral crafts comprise a few big ships and many small boats. The neutral crafts move on fixed randomly selected paths around the Port of Oakland. The patrol helicopter and crafts also move on fixed patrol paths, and in addition, they are able to intercept the small-boat threats by using dynamic pathfinding to find new paths. All the blue entities are able to sense both neutral crafts and threats. The neutral crafts do not have sensing capabilities. The patrol helicopter and crafts also have a sensor to detect whether the threats have entered the intercept range. Big ships have a very large

relative radar cross section compared to small boats. The small boats, in turn, have a larger relative radar cross section than the small-boat threats. The following table lists the objects attributes.

Table 5 Objects Attributes

		Navigation	Sensing Range (km)	Intercept Range (km)	Movement (knots)	Radar Cross Section (relative units)
Blue Entities	1 Patrol Helicopter	●Fixed patrol route ●Interception with dynamic pathfinding	2	0.1	150	Nil
	4 Patrol Crafts	●Different fixed patrol route for each craft ●Interception with dynamic pathfinding	1	0.1	●10 (patrol) ●45 (max)	Nil
	1 Stationary Radar	Fixed location	2.4	Nil	Nil	Nil
	3 High Value Zones	Fixed location	0.5	Nil	Nil	Nil
Neutral Crafts	Few Big Ships	Randomly chosen from fixed routes	Nil	Nil	5	1000
	Many Small Boats	Randomly chosen from fixed routes	Nil	Nil	20	5

4. Operations

The operations in the small-boat threat MAS model consist of sensing, navigation, small-boat-threat activities, blue-entities activities, and results of activities.

a. Sensing

Blue entities and small-boat threats identify a known threat through the detection→classified→recognised→identified cycle. Each phase in the cycle takes three minutes. The blue entities have a 20% chance of detecting arms and marking the target as a known threat in the recognised phase. Identification of the small-boat threats is assured when the identified phase is completed. The

threats identify the blue entities earlier and this occurs on completion of the recognized phase. The cycle is terminated when the target exits the sensing range. The cycle is restarted when the target re-enters the sensing range.

A blue entity can only sense whether a neutral craft or small-boat threat enters or exits a high-value zone if the entity is within the blue-entity's sensing radius. A blue entity can only sense whether a small-boat threat has entered the blue entity's intercept zone if it has been commanded to intercept the target.

Blue entities and small-boat threats start tracking the velocity of an unknown threat when it is first detected. If the velocity exceeds the legal limit, the target is marked as a known threat. Note that when the patrol craft is cruising during its patrol round, its velocity is within the legal limit and is not immediately marked as a threat because of this.

Neutral crafts, such as big ships and small boats, and small-boat threats are deemed to be within cover when they are less than 5m apart. In cover, the craft with the largest radar cross section is detected while the other crafts are lost. If crafts have the same radar cross section, the first cover detecting the other entity is chosen to be detected while the others are hidden.

b. Navigation

The movement of all entities uses uniform linear velocity without acceleration, and this is subject to the maximum speed specified for the entity.

Navigation within the Port of Oakland's generated-pathfinding map results in navigation to any point within the sea mass, and there is no movement onto land.

To reduce complexity in implementation, the patrol helicopter also uses the same navigation map for its dynamic pathfinding. Thus, for interception, the patrol helicopter will move over the sea mass only, skirting any coasts that may obtrude.

c. Small-Boat-Threat Activities

(1) Attacking. Small-boat threats start with attack activity and move at maximum speed to hit their targets.

(2) Hiding. The small-boat threats' hiding means finding, moving to, and following the nearest cover with a large relative radar cross section.

(3) Evasion. Evasion can only result from the near proximity of a blue entity, and results in the small-boat threat's moving generally opposite the direction of the blue entity, at maximum speed.

(4) Escape. Escape can only result from the near proximity of a blue entity, and results in the small-boat threat's returning to its starting location at top speed.

d. Blue-Entity Activities

(1) Intercept and Stop Intercept. Blue entities broadcast significant detections to the blue-entity command center. The command center identifies the blue entity that can engage the target quickest and commands it to intercept.

The blue-entity command center has a priority list of unassigned targets, from known-threats detection to unknown threats in high-value zones, to known threats in high-value zones (which have highest priority).

The blue entity broadcasts the surrender of small-boat threats to the command center. The command center calls off the intercept, and the blue entity resumes patrol.

When the blue entity loses sight of a threat, it broadcasts to inform the command center. If no blue entity can see the threat, the command center calls off any intercepting blue entity.

(2) Non-Lethal or Lethal. A blue entity applies non-lethal or lethal action on a small-boat threat immediately, once it is within intercept range.

e. Results of Activities

(1) Surrender. The small-boat threat immediately surrenders upon non-lethal or lethal action by helicopter or patrol craft.

(2) Successful Attack. The small-boat threat's attack is deemed successful when it reaches its attack target.

(3) Successful Escape. The small-boat threat's escape is deemed successful when it returns to its starting location.

(4) Leaving the Simulation. Neutral crafts leave the simulation on arriving at their destinations and small-boat threats leave the simulation upon a successful attack, escape, or surrender. This prevents false detections of neutral crafts and nonexistent small-boat threats.

Blue entities do not leave the scenario; they continue to monitor and patrol the port.

5. Laws

a. Two-Dimensional World

Both sensing and navigation are performed in a two-dimensional world.

b. Sensor Laws

Both blue entities and small-boat threats can sense multiple targets simultaneously.

The legal speed limit in port waters is 30 knots, and any entity traveling faster, such as the small-boat threat, is marked by tracking sensors as a threat.

c. Movement Laws

Entities will not collide with the land mass; however, multiple entities are allowed to be at the same location in the sea area, as they are assumed likely to reposition themselves to avoid collisions.

d. Activities Laws

A blue entity can intercept only one small-boat threat at a time and is meanwhile unavailable for other assignments. The assignment of the attack target of the small-boat threats is fixed and does not change during the simulation.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. EXPERIMENT, RESULTS AND ANALYSIS

A. EXPERIMENT

1. Hypothesis and Measurements

The null hypothesis is that there is no statistically significant change in the adaptability of the MAS compared to the standard behavior.

The alternate hypothesis is that applying a two-layer, personality-action MAS to small-boat threats increases their adaptability, as compared to a standard implementation.

The parametric t-Test with unequal variances, and the non-parametric Wilcoxon/Kruskal-Wallis (Rank Sums) test was applied, with two-tail probability, using JMP 7 software. As the measurements are dependent, significance remains indicated by an Alpha of 0.05.

Adaptability was measured in the following ways:

i) increase in complexity of operations

This measure was collected from the sub-measures of

a. increase in length of operation time

Null Hypothesis NH1_1: There is no significant change in operation time.

$$Operation\ Time_{Standard} = Operation\ Time_{MAS}$$

Alternate Hypothesis AH1_1: There is significant change in operation time.

b. increase in number of operations.

Null Hypothesis NH1_2: There is no significant change in number of operations.

$$Number\ of\ Operations_{Standard} = Number\ of\ Operations_{MAS}$$

Alternate Hypothesis AH1_2: There is significant change in number of operations.

ii) increase in flexibility of operations

This measure corresponds to an increase in the distribution of actions. This measure was collected from the sub-measures of

a. Attack Activity Count

Null Hypothesis NH2_1: There is no significant change in the attack activity count.

$$Attack\ Activity\ Count_{Standard} = Attack\ Activity\ Count_{MAS}$$

Alternate Hypothesis AH2_1: There is significant change in the attack activity count.

b. Hide Activity Count

Null Hypothesis NH2_2: There is no significant change in the hide activity count.

$$Hide\ Activity\ Count_{Standard} = Hide\ Activity\ Count_{MAS}$$

Alternate Hypothesis AH2_2: There is significant change in the hide activity count.

c. Evade Activity Count

Null Hypothesis NH2_3: There is no significant change in the evade activity count.

$$Evade\ Activity\ Count_{Standard} = Evade\ Activity\ Count_{MAS}$$

Alternate Hypothesis AH2_3: There is significant change in the evade activity count.

d. Escape Activity Count

Null Hypothesis NH2_4: There is no significant change in the escape activity count.

$$Escape\ Activity\ Count_{Standard} = Escape\ Activity\ Count_{MAS}$$

Alternate Hypothesis AH2_4: There is significant change in the escape activity count.

iii) increase in success of operations

This measure corresponds to an increase in successful attacks by the small-boat threats.

Null Hypothesis NH3: There is no significant change in the attack success count.

$$Attack\ Success\ Count_{Standard} = Attack\ Success\ Count_{MAS}$$

Alternate Hypothesis AH3: There is significant change in the attack success count.

2. Experimental Setup

The following screenshot shows the simulation application used for this experiment.

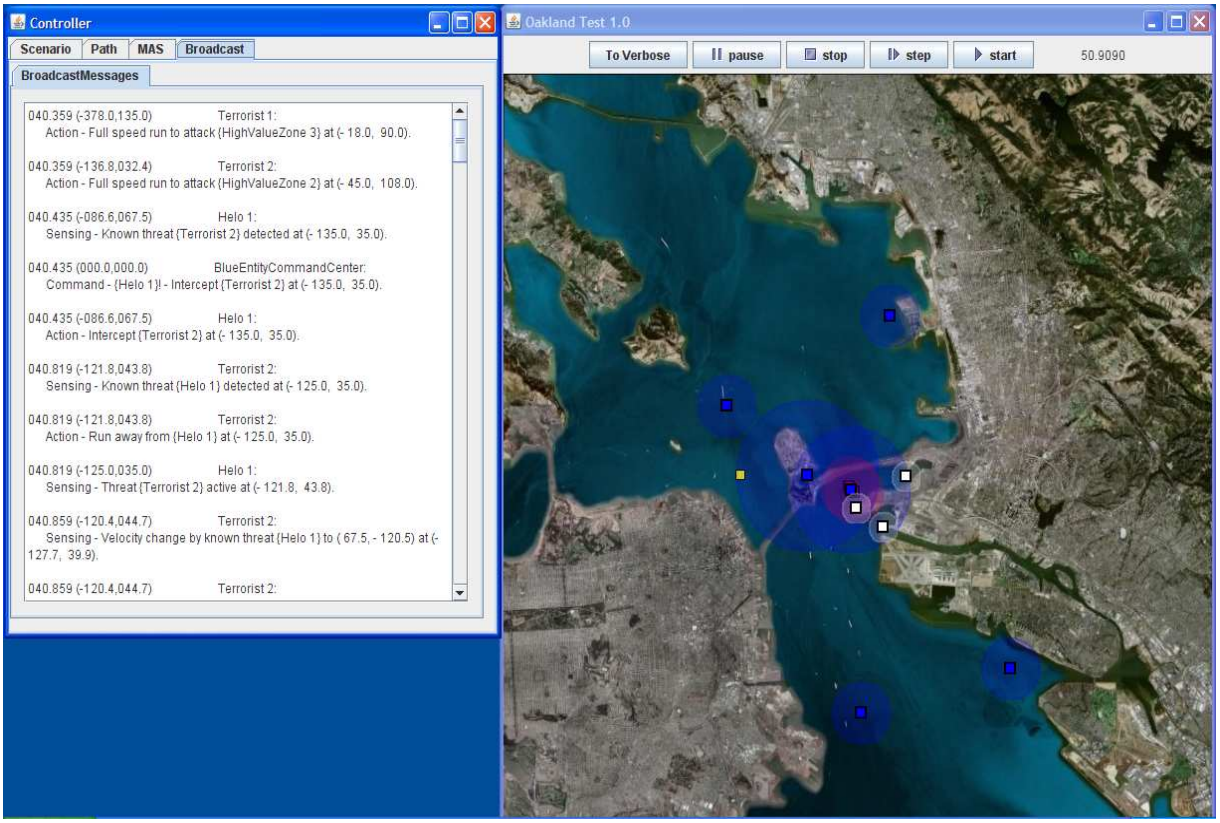


Figure 31 Screen Capture of Simulation Application

The simulation was executed with thirty runs using standard behavior. The simulation was then executed with thirty runs using MAS behavior.

The following data was collected for each run:

i) operation time

The time from first arrival to completion of attack by four small-boat threats.

ii) number of activities per simulation run

The total number of activities executed by the four threats per simulation run.

iii) attack-activity count

The total number of attack activities executed by the four threats per simulation run.

iv) hide-activity count

The total number of hides executed by the four small-boat threats per simulation run.

v) evade-activity count

The total number of evade activities executed by the four small-boat threats per simulation run.

vi) escape-activity count

The total number of escape activities executed by the four small-boat threats per simulation run.

vii) attack-success count

The total number of successful attacks by the four small-boat threats per simulation run.

The following picture shows a screen snapshot of the measurement data output by the simulation application.

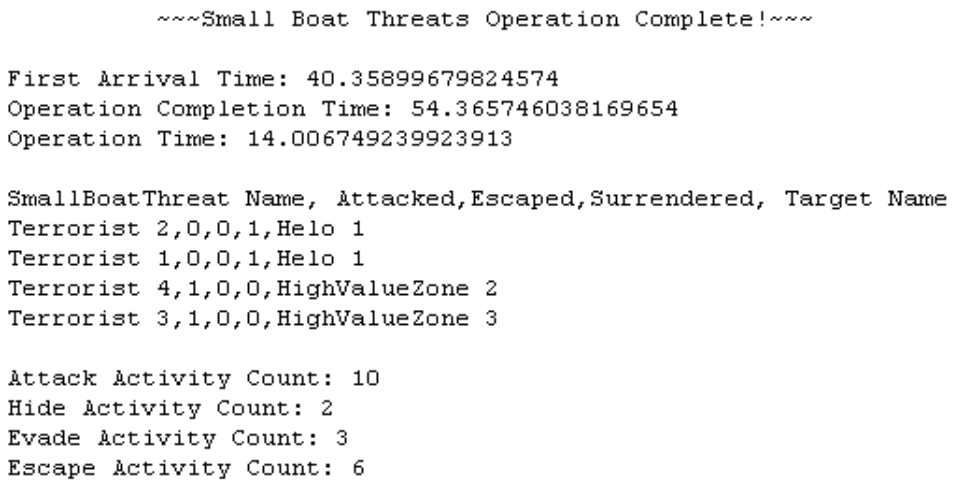


Figure 32 Screen Snapshot of Measurement Data Output by Simulation Application

The following picture shows a screen snapshot of the MAS data output by the simulation application. Note that the some of the action weights are modified during the course of the simulation.

```
Terrorist 2 MAS Personality Action Weights (Attack,Hide,Evade,Escape)
Suicide Personality Type: (0.2954,0.2341,0.2361,0.2344)
Tactical Personality Type: (0.2608,0.2608,0.2608,0.2176)
Deceptive Personality Type: (0.2503,0.2497,0.2497,0.2503)
Balanced Personality Type: (0.2500,0.2500,0.2500,0.2500)

Terrorist 1 MAS Personality Action Weights (Attack,Hide,Evade,Escape)
Suicide Personality Type: (0.2989,0.2329,0.2325,0.2357)
Tactical Personality Type: (0.2012,0.2704,0.2704,0.2579)
Deceptive Personality Type: (0.2499,0.2504,0.2499,0.2499)
Balanced Personality Type: (0.2500,0.2500,0.2500,0.2500)

Terrorist 4 MAS Personality Action Weights (Attack,Hide,Evade,Escape)
Suicide Personality Type: (0.2500,0.2500,0.2500,0.2500)
Tactical Personality Type: (0.2500,0.2500,0.2500,0.2500)
Deceptive Personality Type: (0.2500,0.2500,0.2500,0.2500)
Balanced Personality Type: (0.2500,0.2500,0.2500,0.2500)

Terrorist 3 MAS Personality Action Weights (Attack,Hide,Evade,Escape)
```

Figure 33 Screen Snapshot of MAS Data Output by Simulation Application

B. RESULTS

The following statistical results are compiled from the collected measurements. Refer to Appendix A: Results of Measurements and Statistical Results for details.

Table 6 Summary of Statistical Results

		Operation Time	Total Activity Count	Attack Activity Count	Hide Activity Count	Evade Activity Count	Escape Activity Count	Attack Success Count
Std	Mean	12.93434	16.96667	14.5	0.133333	2.333333	0	2.166667
	Standard Deviation	2.479213	7.289498	6.112452	0.571346	2.264164	0	0.791478
MAS	Mean	13.04888	17.53333	8.066667	0.566667	5.5	3.4	2.466667
	Standard Deviation	2.348321	8.985557	2.899861	1.406471	4.040741	2.823546	0.937102

		Operation Time	Total Activity Count	Attack Activity Count	Hide Activity Count	Evade Activity Count	Escape Activity Count	Attack Success Count
	Change in Mean from Std to MAS	**0.11**	**0.57**	**-6.43**	**0.43**	**3.17**	**3.40**	**0.30**
	t Test with Unequal Variances, Prob > |t|:	**0.8549**	**0.7895**	**<0.0001**	**0.1262**	**0.0005**	**<0.0001**	**0.1857**
	Wilcoxon / Kruskal-Wallis Test (Rank Sums), Prob > |Z|	**0.8882**	**0.7839**	**<0.0001**	**0.1268**	**0.0003**	**<0.0001**	**0.2444**

C. ANALYSIS

1. Assessment of Complexity of Operations

The null hypotheses NH1_1 “There is no significant change in operation time” and NH1_2 “There is no significant change in number of operations” failed to be rejected. Thus, there is no statistical significance that MAS behavior increased operation time and total activity count.

2. Assessment of Flexibility of Operations

The null hypotheses NH2_1 “There is no significant change in the attack activity count”, NH2_3 “There is no significant change in the evade activity count” and NH2_4 “There is no significant change in the escape activity count” are rejected. The null hypothesis NH2_2 “There is no significant change in the hide activity count” failed to be rejected.

As the figures are related and there is a greater distribution of activity to evade and escape activities, it is concluded that there is statistical significance that MAS behavior increased utility of a variety of operations.

3. Assessment of Success of Operations

The null hypothesis NH2_1 “There is no significant change in the attack success count” failed to be rejected. There is no statistical significance that MAS behavior increased success of operations.

4. Observed Artifacts

There are zero cases of successful escapes, although there are recorded attempts of escape activity made by the threats.

5. Assessment of Whether MAS Behavior Improved Adaptability of Small-Boat Threats

There is statistical significance that the implemented MAS behavior increased the flexibility of operations, but there is no statistical significance that the implemented MAS behavior increased the complexity of the operations and the success of operations.

Reviewing the summary of results reveals that there are increases in the mean for the measurements for the complexity of operation and the success of operations. These increases are relatively small for both measurements for complexity of operations, but larger for success of operations. In the latter case, there is a favorable chance that the implemented MAS behavior improved the complexity of operations and especially the success of operations.

It can thus be concluded that MAS behavior demonstrates improvement of the operational flexibility and shows potential for improving the adaptability of the small-boat threats.

V. CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK

A. CONCLUSIONS

Discrete-event simulation with Simkit facilitated the creation of autonomous and interactive sensing agents in a maritime time–space environment. Dynamic pathfinding improved the flexibility of dynamic asymmetric threats and maritime assets in finding their way to their targets. With the implementation of MAS behavior, asymmetric threats demonstrate greater flexibility of behaviors, show slight improvement in success and complexity of operations, and evince potential for improving adaptability. In maritime security, dynamic asymmetric threats will enable the simulation of a wider variety of maritime threat scenarios and play an important part in improving plans for maritime force and infrastructure configurations.

B. RECOMMENDATIONS

MAS behaviors can be used to enhance hard-coded behaviors in simulations by assigning them with personalities, decisions, and action choices. While still conforming to the rules and boundaries of the environment, these MAS behaviors enable the agents to interact with the environment in a dynamic and flexible manner. The agents have more free play to explore the environment with flexible behaviors, and the variety of potential situations in the simulation will increase, expanding simulation exploration into previously uncharted waters and increasing their value.

MAS behaviors enable agents to switch personality contexts according to situation, allowing them to choose appropriate actions. With adaptability, agents may adjust to changing situations, allowing them more maneuverability,

survivability, and success in achieving operational goals. Dynamic asymmetric threats will pose a more potent threat in simulations and facilitate wider explorations of force- and defense-infrastructure configurations.

C. FUTURE WORK

Potential future work includes:

i) Study interactions and refine maritime scenarios with stakeholders to identify potential tactics and improvements in maritime-defense infrastructure,

ii) Explore application of the genetic algorithm to the natural evolution of personality genes, to improve chances of successful attack,

iii) Explore the application of cognitive blending to blend inputs and improve decision making,

iv) Explore application of MAS behavior to blue entities and the affected changes in interaction between the agents,

v) Explore changes in interaction of the agents with human-controlled agents, and

vi) Explore integration with Terence Tan's intelligent blue entities [20] to study the interactions between red and blue intelligent agents.

APPENDIX A: RESULTS OF MEASUREMENTS AND STATISTICAL RESULTS

The following table shows the results of the thirty runs for standard behavior.

Table 7 Results of Thirty Runs for Standard Behavior

S\N	Operation Time	Attack Success Count	Attack Activity Count	Hide Activity Count	Evade Activity Count	Escape Activity Count	Total Activity Count
1	10.270799	2	11	3	2	0	16
2	10.716136	1	21	0	1	0	22
3	13.630137	3	17	0	2	0	19
4	11.662902	3	19	0	1	0	20
5	17.001678	2	13	0	5	0	18
6	10.302817	3	20	0	0	0	20
7	10.129052	2	28	0	9	0	37
8	14.785756	3	9	0	5	0	14
9	12.17068	2	23	0	9	0	32
10	10.503485	1	9	0	2	0	11
11	11.793807	3	4	0	2	0	6
12	10.503485	2	21	0	4	0	25
13	16.793807	1	9	0	2	0	11
14	10.503485	1	9	0	2	0	11
15	14.785756	2	18	0	2	0	20
16	14.785756	3	6	0	0	0	6
17	10.916609	0	17	1	2	0	20
18	14.785756	3	14	0	4	0	18
19	12.17068	3	19	0	1	0	20
20	10.143368	2	5	0	1	0	6
21	14.006749	2	9	0	2	0	11
22	14.785756	2	19	0	1	0	20
23	8.50604	2	18	0	2	0	20
24	14.228626	3	4	0	0	0	4
25	14.256365	2	15	0	3	0	18
26	16.793807	3	10	0	2	0	12
27	15.588526	2	18	0	1	0	19
28	10.654575	2	18	0	0	0	18
29	17.001678	3	13	0	0	0	13
30	13.852014	2	19	0	3	0	22
Sum	388.03008	65	435	4	70	0	509
Mean	**12.934336**	**2.1666667**	**14.5**	**0.1333333**	**2.3333333**	**0**	**16.9667**

	S\N	Operation Time	Attack Success Count	Attack Activity Count	Hide Activity Count	Evade Activity Count	Escape Activity Count	Total Activity Count
Standard Deviation		2.4792126	0.7914776	6.112452	0.5713465	2.2641636	0	7.2895
95% Confidence Interval		**0.8871585**	**0.2832214**	**2.1872727**	**0.2044499**	**0.8102056**	**0**	**0**
Variance		6.1464952	0.6264368	37.362069	0.3264368	5.1264368	0	53.1368

The following table shows the results of the thirty runs for MAS behavior.

Table 8 Results of Thirty Runs for MAS Behavior

S\N	Operation Time	Attack Success Count	Attack Activity Count	Hide Activity Count	Evade Activity Count	Escape Activity Count	Total Activity Count
1	11.141019	3	4	0	1	2	7
2	9.8760498	2	5	1	6	1	13
3	13.852014	3	10	0	13	2	25
4	10.129052	4	6	0	4	4	14
5	15.635929	3	5	0	1	0	6
6	10.302817	3	11	1	6	3	21
7	10.129052	2	11	3	16	8	38
8	9.8760498	3	7	0	5	2	14
9	14.234592	2	5	0	5	1	11
10	14.785756	1	8	4	2	3	17
11	13.394504	1	8	0	9	7	24
12	12.17068	2	15	0	13	7	35
13	10.729634	1	5	0	3	1	9
14	11.141019	2	8	0	1	0	9
15	14.006749	4	8	0	9	4	21
16	14.006749	4	8	6	5	5	24
17	15.778072	2	10	0	8	3	21
18	16.625066	3	7	0	3	1	11
19	12.135078	3	9	0	9	9	27
20	10.143368	4	7	0	4	3	14
21	9.8760498	3	4	0	1	1	6
22	13.080629	1	8	0	2	2	12
23	16.793807	2	7	0	2	1	10
24	16.625066	3	5	0	1	1	7
25	13.852014	3	6	0	5	3	14
26	12.17068	3	15	0	12	11	38
27	15.588183	1	12	0	4	6	22
28	16.625066	2	7	0	6	1	14
29	12.754994	2	11	0	6	4	21
30	14.006749	2	10	2	3	6	21

	S\N	Operation Time	Attack Success Count	Attack Activity Count	Hide Activity Count	Evade Activity Count	Escape Activity Count	Total Activity Count
Sum		391.46648	74	242	17	165	102	526
Mean		**13.048883**	**2.4666667**	**8.0666667**	**0.5666667**	**5.5**	**3.4**	**17.5333**
Standard Deviation		2.3483209	0.9371024	2.8998613	1.4064711	4.0407408	2.8235463	8.98556
95% Confidence Interval		**0.8403204**	**0.3353316**	**1.037683**	**0.50329**	**1.445934**	**1.0103745**	**0**
Variance		5.5146109	0.8781609	8.4091954	1.9781609	16.327586	7.9724138	80.7402

The follow pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the operation time measurement.

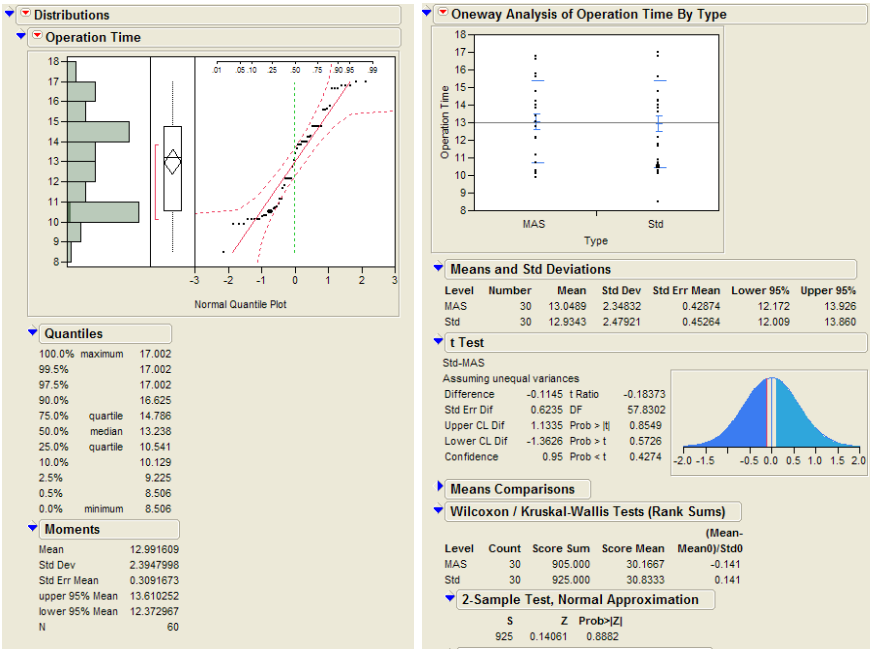


Figure 34 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Operation Time

The following pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the total activity-count measurement.

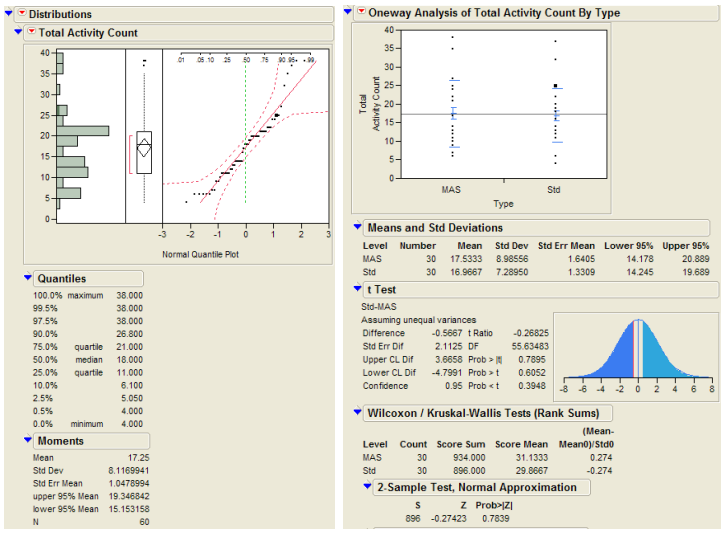


Figure 35 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Total Activity Count

The follow pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the attack-activity-count measurement.

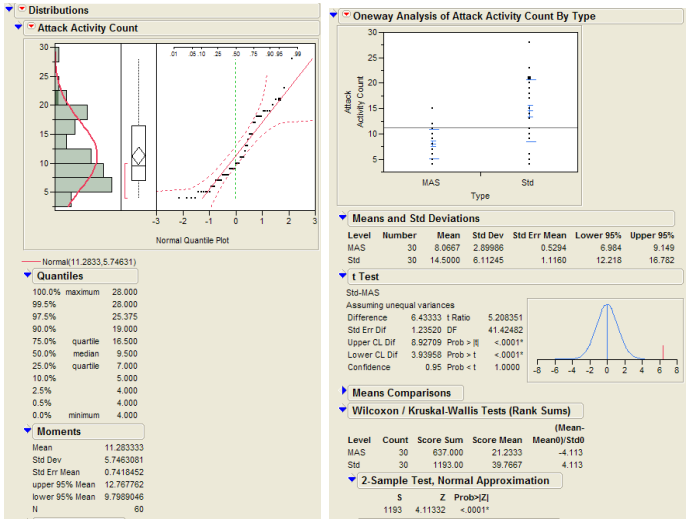


Figure 36 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Attack-Activity Count

The follow pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the hide-activity-count measurement.

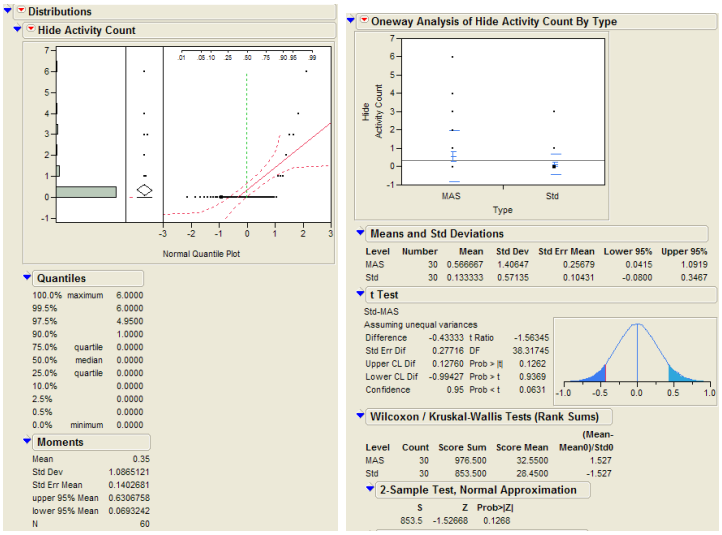


Figure 37 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Hide-Activity Count

The follow pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the evade-activity-count measurement.

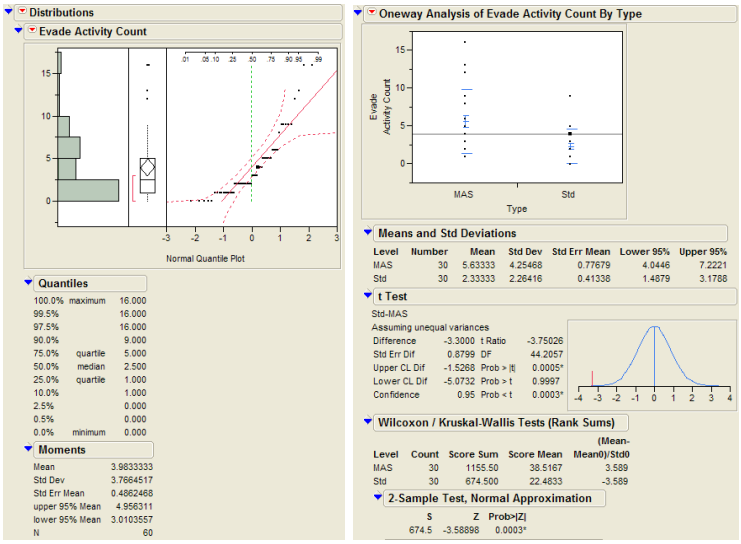


Figure 38 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Evade-Activity Count

The follow pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the escape-activity-count measurement.

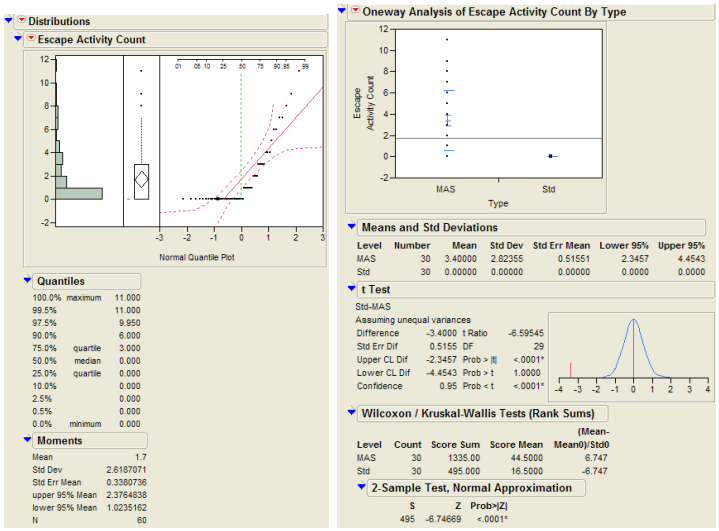


Figure 39 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Escape-Activity Count

The follow pictures show the screen snapshots of the distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) test of the successful-attack-count measurement.

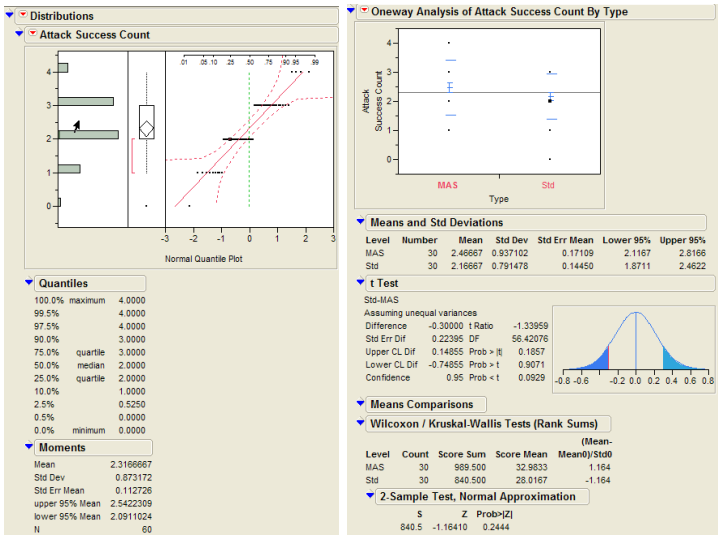


Figure 40 Distribution, t-Test, and Wilcoxon/Kruskal-Wallis (Rank Sums) Test of Successful-Attack Count

APPENDIX B: CODE SNIPPETS

```
//=====================================================
// Snippets from PortCookieCutterMediator.java
//=====================================================
    /**
     * Mediates between sensor and cover to enforce cover scenarios
     * Use PortContact instead of Contact to enable access to source of contact
     */
    public void doEnterRange(Sensor sensor, Mover target) {
        if (this == SensorTargetMediatorFactory.getInstance().getMediatorFor(
                sensor.getClass(), target.getClass())) {
            if (sensor instanceof PortCoverSensor) {
                Mover mover = sensor.getMover();
                if ((mover instanceof RadarCrossSection) && (target instanceof RadarCrossSection)) {
                    double sensorRadarCrossSection = ((RadarCrossSection)mover).getRadarCrossSection();
                    double targetRadarCrossSection = ((RadarCrossSection)target).getRadarCrossSection();
                    if ((targetRadarCrossSection < sensorRadarCrossSection) ||
                        ((Math.abs(targetRadarCrossSection - sensorRadarCrossSection)
                        < Double.MIN_VALUE) && !isCoveredBy(mover, target))) {
                        // set covered target
                        coveredTargets.add(target);
                        covers.put(target, mover);
                        notifySensorsTargetUndetected(target);
                    }
                }
            } else {
                if (!isCoveredTarget(target)) {
                    Contact contact = contacts.get(target);
                    if (contact == null) {
                        contact = new PortContact(target);
                        contacts.put(target, contact);
                    }
                    sensor.waitDelay("Detection", 0.0, contact);
                }
            }
        }
    }

    /**
     * Mediates between sensor and cover to enforce cover scenarios
     * Create PortContact to ensure that a non-null contact is passed to event
     */
    public void doExitRange(Sensor sensor, Mover target) {
        if (this == SensorTargetMediatorFactory.getMediator(
                sensor.getClass(), target.getClass())) {
            if (sensor instanceof PortCoverSensor) {
                Mover mover = sensor.getMover();
                if ((mover instanceof RadarCrossSection) && (target instanceof RadarCrossSection)) {
                    double sensorRadarCrossSection = ((RadarCrossSection)mover).getRadarCrossSection();
                    double targetRadarCrossSection = ((RadarCrossSection)target).getRadarCrossSection();
                    if ((targetRadarCrossSection < sensorRadarCrossSection) ||
                        ((Math.abs(targetRadarCrossSection - sensorRadarCrossSection)
                        < Double.MIN_VALUE) && isCoveredBy(target, mover))) {
                        // uncover target
                        coveredTargets.remove(target);
                        // remove cover
                        covers.remove(target);
                        notifySensorsTargetDetected(target);
                    }
                }
            } else {
                if (!isCoveredTarget(target)) {
                    Contact contact = contacts.get(target);
                    if (contact == null) {
```

```
                contact = new PortContact(target);
                contacts.put(target, contact);
            }
            sensor.waitDelay("Undetection", 0.0, contact);
        }
    }
  }
}
//=========================================================
// End Snippets from PortCookieCutterMediator.java
//=========================================================


//=========================================================
// Snippets from SmallBoatThreat.java
//=========================================================
    /**
     * respond to broadcasts of threat detected
     */
    public void doBroadcast(BroadcastMessage message) {
        if (this.isLeave) {
            return;
        }

        // if successful, stop activities
        if ((message.getBroadcastMessageType().compareTo(BroadcastMessageType.SUCCESSFUL_ATTACK) == 0) ||
            (message.getBroadcastMessageType().compareTo(BroadcastMessageType.SUCCESSFUL_ESCAPE) == 0)) {
            // set leaving to true
            this.isLeave = true;
            // stop activities
            stopActivities();
            return;
        }

        // if caught, surrender
        if (isCaught(message)) {
            surrender((SimEntity)message.getParameters()[0]);
            return;
        }

        // prevent deadlock in repeated activities
        if (this.checkRepeatAction()) {
            return;
        }

        // run behaviors
        if (RunOaklandPort.getUseMAS()) {
            runMASBehavior(message);
        } else {
            runStandardBehavior(message);
        }
    }

    /**
     * MAS behavior
     */
    protected void runMASBehavior(BroadcastMessage message) {
        BroadcastMessage decision = checkMASDecision(message);
        if (decision.getBroadcastMessageType().equals(BroadcastMessageType.SURRENDER)) {
            surrender((SimEntity)decision.getParameters()[0]);
        } else if (decision.getBroadcastMessageType().equals(BroadcastMessageType.ATTACK)) {
            attack();
        } else if (decision.getBroadcastMessageType().equals(BroadcastMessageType.HIDE)) {
            hide((SimEntity)decision.getParameters()[0]);
        } else if (decision.getBroadcastMessageType().equals(BroadcastMessageType.EVADE)) {
            evade((SimEntity)decision.getParameters()[0]);
        } else if (decision.getBroadcastMessageType().equals(BroadcastMessageType.ESCAPE)) {
            escape((SimEntity)decision.getParameters()[0]);
```

```
    }
}

/**
 * Standard behavior
 */
protected void runStandardBehavior(BroadcastMessage message) {
    if (isChased()) {
        SimEntity coverEntity = null;
        if (canHitTargetFirst(message)) {
            attack();
        } else if ((coverEntity = findNearestCover(message)) != null) {
            hide(coverEntity);
        } else if (canOutrun(message)) {
            evade((SimEntity)message.getParameters()[0]);
        } else {
            escape((SimEntity)message.getParameters()[0]);
        }
    } else {
        attack();
    }
}

// number of personality types
private int masNumberOfPersonalityTypes = 4;

// number of action types
private int masNumberOfActionTypes = 4;

// 0 = suicide personality
// 1 = tactical personality
// 2 = deceptive personality
// 3 = balanced personality
// {motivation for attack=reduce distance to attack target,
// motivation for tactical=increase distance from nearest blue entity,
// motivation for deception=reduce distance to nearest cover}
private double[][] masPersonality = new double[][] {
    {1.0, 0.0, 0.0}, // suicide
    {0.0, 1.0, 0.0}, // tactical
    {0.0, 0.0, 1.0}, // deceptive
    {0.3, 0.3, 0.3} // balanced
};

// {attack, hide, evade, escape}
private double[][] masActionWeights = new double[][] {
    {0.25, 0.25, 0.25, 0.25}, // suicide
    {0.25, 0.25, 0.25, 0.25}, // tactical
    {0.25, 0.25, 0.25, 0.25}, // deceptive
    {0.25, 0.25, 0.25, 0.25} // balanced
};

// history of happiness
private List<Double> masDecisionHistoryHappiness = new ArrayList<Double>();

// history of decisions
private List<BroadcastMessage> masDecisionHistory = new ArrayList<BroadcastMessage>();

// history of decision times
private List<Double> masDecisionHistoryTimes = new ArrayList<Double>();

// history of personalities
private List<Integer> masDecisionHistoryPersonality = new ArrayList<Integer>();

// history of distance to attack target
private List<Double> masDecisionHistoryDistanceToAttackTarget = new ArrayList<Double>();

// history of distance to nearest blue entity
private List<Double> masDecisionHistoryDistanceToNearestBlueEntity = new ArrayList<Double>();
```

```
// history of distance to nearest cover
private List<Double> masDecisionHistoryDistanceToNearestCover = new ArrayList<Double>();

/**
 * Decisions can be to attack, hide, evade or escape.
 * Decisions are encoded and returned as a broadcast message.
 * Inputs to MAS include:
 * Goals are
 * i) attack target, ii) avoid being intercepted, iii) find cover
 * Methods are
 * i) Fixed personality weights, ii) Changing personality+action probabilities
 * Happiness is based on
 * i) proximity to attack target, ii) distance away from blue entity,
 * iii) distance away from cover
 * Actions are
 * i) attack, ii) hide, iii) evade, iv) escape
 * Conditions are
 * i) whether cover is nearby ii) whether blue entity is nearby
 * Personalities are
 * i) suicide (favors reducing proxmity to attack target),
 * ii) tactical (favors increasing distance away from blue entity)
 * iii) deceptive (facors reducing distance to cover)
 * iv) balanced (balance)
 */
public BroadcastMessage checkMASDecision(BroadcastMessage message) {

    // get current decision time
    double currentDecisionTime = Schedule.getSimTime();

    // get distance to attack target
    Point2D location = this.getMover().getLocation();
    double distanceToAttackTarget =
            location.distance(this.attackTarget.getLocation());

    // get distance to nearest blue entity
    double distanceToNearestBlueEntity = this.maxRange;
    BlueEntity nearestBlueEntity = null;
    for (Entry<SimEntity, Boolean> entry : contacts.entrySet()) {
        if (entry.getValue().booleanValue()) { // if known threat
            if (entry.getKey() instanceof BlueEntity) {
                BlueEntity blueEntity = (BlueEntity)entry.getKey();
                double newDistance = location.distance(blueEntity.getLocation());
                if (newDistance < distanceToNearestBlueEntity) {
                    distanceToNearestBlueEntity = newDistance;
                    nearestBlueEntity = blueEntity;
                }
            }
        }
    }

    // get distance to nearest cover
    double distanceToNearestCover = this.maxRange;
    Map<Moveable, Double> nearestEntities =
            port.findNearestEntities(getMover(), this.maxRange);
    Moveable nearestEntity = null;
    for (Entry<Moveable, Double> entry : nearestEntities.entrySet()) {
        Moveable moveable = entry.getKey();
        if (moveable == this) { // avoid choosing itself
            continue;
        }
        double newDistance =  location.distance(moveable.getLocation());
        if (newDistance < distanceToNearestCover) {
            distanceToNearestCover = newDistance;
            nearestEntity = moveable;
        }
    }
```

```
// get last decision
BroadcastMessage lastDecision = null;
if (!this.getMasDecisionHistory().isEmpty()) {
   lastDecision = getMasDecisionHistory().get(getMasDecisionHistory().size() - 1);
}

// get last decision time
double lastDecisionTime = currentDecisionTime;
if (!this.getMasDecisionHistoryTimes().isEmpty()) {
   lastDecisionTime =
         getMasDecisionHistoryTimes().get(
         getMasDecisionHistoryTimes().size() - 1).doubleValue();
}

// get last distance to attack target
double lastDistanceToAttackTarget = Double.MAX_VALUE;
if (!this.getMasDecisionHistoryDistanceToAttackTarget().isEmpty()) {
   lastDistanceToAttackTarget =
         getMasDecisionHistoryDistanceToAttackTarget().get(
         getMasDecisionHistoryDistanceToAttackTarget().size() - 1).doubleValue();
}

// get last distance to nearest blue entity
double lastDistanceToNearestBlueEntity = Double.MAX_VALUE;
if (!this.getMasDecisionHistoryDistanceToNearestBlueEntity().isEmpty()) {
   lastDistanceToNearestBlueEntity =
         getMasDecisionHistoryDistanceToNearestBlueEntity().get(
         getMasDecisionHistoryDistanceToNearestBlueEntity().size() - 1).doubleValue();
}

// get last distance to nearest cover
double lastDistanceToNearestCover = Double.MAX_VALUE;
if (!this.getMasDecisionHistoryDistanceToNearestCover().isEmpty()) {
   lastDistanceToNearestCover =
         getMasDecisionHistoryDistanceToNearestCover().get(
         getMasDecisionHistoryDistanceToNearestCover().size() - 1).doubleValue();
}

// get last personality
int lastPersonalityType = 0;
if (!this.getMasDecisionHistoryPersonality().isEmpty()) {
   lastPersonalityType =
         getMasDecisionHistoryPersonality().get(
         getMasDecisionHistoryPersonality().size() - 1).intValue();
}

// get last last decision happiness
double lastLastDecisionHappiness = 0.0;
if (!this.getMasDecisionHistoryHappiness().isEmpty()) {
   lastLastDecisionHappiness =
         getMasDecisionHistoryHappiness().get(
         getMasDecisionHistoryHappiness().size() - 1).doubleValue();
}

// compute last decision happiness
double maxAttackDistance = startingLocation.distance(attackTarget.getLocation());
double normChangeInAttackDistance =
      ((lastDistanceToAttackTarget == Double.MAX_VALUE) ? 0 :
         (lastDistanceToAttackTarget - distanceToAttackTarget)) / maxAttackDistance;
double normChangeInDistanceToNearestBlueEntity =
      ((lastDistanceToNearestBlueEntity == Double.MAX_VALUE) ? 0 :
         (distanceToNearestBlueEntity - lastDistanceToNearestBlueEntity)) / this.maxRange;
double normChangeInDistanceToNearestCover =
      ((lastDistanceToNearestCover == Double.MAX_VALUE) ? 0 :
         (lastDistanceToNearestCover - distanceToNearestCover)) / this.maxRange;
double lastDecisionHappiness =
      (normChangeInAttackDistance * getMasPersonality()[lastPersonalityType][0] +
      normChangeInDistanceToNearestBlueEntity * getMasPersonality()[lastPersonalityType][1] +
```

```
        normChangeInDistanceToNearestCover * getMasPersonality()[lastPersonalityType][2]) / 3;

// update history for happiness
if (lastDecision != null) { // if a decision has been made previously
    // update the decision happiness
    this.getMasDecisionHistoryHappiness().add(new Double(lastDecisionHappiness));

    // update the action weights
    int lastActionType = this.getActionType(lastDecision.getBroadcastMessageType());
    for (int i = 0; i < this.getMasNumberOfActionTypes(); i++) {
        if (i == lastActionType) {
            // give half of the happiness to the last action type, and normalize
            this.getMasActionWeights()[lastPersonalityType][i] =
                    (this.getMasActionWeights()[lastPersonalityType][i] +
                    (0.5 * lastDecisionHappiness)) / (1.0 + lastDecisionHappiness);
        } else {
            // distribute the remaining happiness among the rest of the action types, and normalize
            this.getMasActionWeights()[lastPersonalityType][i] =
                    (this.getMasActionWeights()[lastPersonalityType][i] +
                    (0.5 * lastDecisionHappiness / (this.getMasNumberOfActionTypes() - 1)))
                    / (1.0 + lastDecisionHappiness);
        }
    }
}

// choose next personality that is likely to give highest happiness
int personality = 0;
double decisionHappiness =
        normChangeInAttackDistance * getMasPersonality()[0][0] +
        normChangeInDistanceToNearestBlueEntity * getMasPersonality()[0][1] +
        normChangeInDistanceToNearestCover * getMasPersonality()[0][2];
for (int i = 1; i < getMasNumberOfPersonalityTypes(); i++) {
    double newDecisionHappiness =
        normChangeInAttackDistance * getMasPersonality()[i][0] +
        normChangeInDistanceToNearestBlueEntity * getMasPersonality()[i][1] +
        normChangeInDistanceToNearestCover * getMasPersonality()[i][2];
    if (newDecisionHappiness > decisionHappiness) {
        decisionHappiness = newDecisionHappiness;
        personality = i;
    }
}

// choose next action that is likely to give highest happiness
double randomValue = Math.random();
double cumulativeProbability = 0.0;
int actionType = 0;
SimEntity coverEntity = null;
for (int i = 0; i < this.getMasNumberOfActionTypes(); i++) {
    cumulativeProbability += this.getMasActionWeights()[personality][i];
    if (randomValue < cumulativeProbability) {
        if (i == 1) { // check condition if cover nearby
            if (nearestEntity != null) {
                double nearestDistance = Double.MAX_VALUE;
                for (Entry<Moveable, Double> entry : nearestEntities.entrySet()) {
                    Moveable moveable = entry.getKey();
                    if (moveable == this) { // avoid choosing itself
                        continue;
                    }
                    // if cover moving in direction of attack target and nearest
                    double newDistance =  location.distance(moveable.getLocation());
                    if (Math2D2.isHeadingTowardsTarget(moveable.getLocation(), moveable.getVelocity(),
                            attackTarget.getLocation(), attackTarget.getVelocity(), Math.PI / 4) &&
                            (newDistance < nearestDistance) && (moveable instanceof SimEntity)) {
                        // choose cover
                        nearestDistance = newDistance;
                        coverEntity = (SimEntity)moveable;
                    }
                }
```

```
                    if (coverEntity != null) {
                        actionType = i;
                        break;
                    }
                }
            } else if ((i == 2) || (i == 3)) { // check condition if blue entity nearby
                if (nearestBlueEntity != null) {
                    actionType = i;
                    break;
                }
            } else {
                actionType = i;
                break;
            }
        }
    }

    // create decision
    BroadcastMessage decision = null;
    switch (actionType) {
        case 0: // attack
        default:
            decision = new BroadcastMessage(this, null, BroadcastMessageType.ATTACK, attackTarget);
            break;
        case 1: // hide
            decision = new BroadcastMessage(this, null, BroadcastMessageType.HIDE, coverEntity);
            break;
        case 2: // evade
            decision = new BroadcastMessage(this, null, BroadcastMessageType.EVADE, nearestBlueEntity);
            break;
        case 3: // escape
            decision = new BroadcastMessage(this, null, BroadcastMessageType.ESCAPE, nearestBlueEntity);
            break;
    }

    // update history for decision
    getMasDecisionHistory().add(decision);

    // update history for decision time
    getMasDecisionHistoryTimes().add(new Double(currentDecisionTime));

    // update history for personality
    this.getMasDecisionHistoryPersonality().add(new Integer(personality));

    // update history for distance to attack target
    this.getMasDecisionHistoryDistanceToAttackTarget().add(new Double(distanceToAttackTarget));

    // update history for distance to nearest blue entity
    this.getMasDecisionHistoryDistanceToNearestBlueEntity().add(new Double(distanceToNearestBlueEntity));

    // update history for distance to nearest cover
    this.getMasDecisionHistoryDistanceToNearestCover().add(new Double(distanceToNearestCover));

    return decision;
  }
//=======================================================
// End Snippets from SmallBoatThreat.java
//=======================================================
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Round Table of International Shipping Association, “Shipping Key Facts Introduction,” http://www.marisec.org/shippingfacts/keyfacts.htm, last accessed November 2, 2007.

[2] International Maritime Organization, “International Shipping - carrier of World Trade. World Maritime Day 2005,” http://www.imo.org/InfoResource/mainframe.asp?topic_id=1562&doc_id=8541, last accessed November 2, 2007.

[3] Systems Engineering and Analysis Cohort 11, “Port Security Strategy 2012,” Meyer Institute of Systems Engineering, Naval Postgraduate School, June 15, 2007.

[4] Matthew D. Childs, “An Exploratory Analysis of Water Front Force Protection Measures Using Simulation,” M.S. thesis, Naval Postgraduate School, March 2002.

[5] James William Harney, “Analyzing Anti-Terrorist Tactical Effectiveness of Picket Boats for Force Protection of Navy Ships using X3D Graphics and Agent-based Simulation,” M.S. thesis, Naval Postgraduate School, March 2003.

[6] Oliver Tan Kok Soon, “A Multi-agent System for Tracking the Intent of Surface Contacts in Ports and Waterways,” M.S. thesis, Naval Postgraduate School, March 2005.

[7] Patrick Joseph Sullivan, “Evaluating the Effectiveness of Waterside Security Alternatives for Force Protection of Navy Ships and Installations using X3D Graphics and Agent-based Simulation,” M.S. thesis, Naval Postgraduate School, September 2006.

[8] Brian A. Jackson et al., “Breaching the fortress wall: understanding terrorist efforts to overcome defensive technologies,” RAND, 2007.

[9] Paul W. Parfomak and John Frittelli, “Maritime Security: Potential Terrorist Attacks and Protection Priorities,” CRS Report RL33787, January 9, 2007.

[10] Wikipedia, “Al-Qaeda,” http://en.wikipedia.org/wiki/Al-Qaeda, last accessed October 10, 2007.

[11] Wikipedia, “Abu Sayyaf,” http://en.wikipedia.org/wiki/Abu_Sayyaf, last accessed October 10, 2007.

[12] National Geospatial-Intelligence Agency, “Anti-Shipping Activity Messages (ASAM),” http://pollux.nss.nima.mil/asam/asam_j_query.html#Search%20the%20ASAM%20Database, last accessed November 2, 2007.

[13] Martin Murphy, “Maritime threat: tactics and technology of the Sea Tigers,” Jane's Intelligence Review, June 1, 2006.

[14] Anthony Davis, “Piracy in Southeast Asia shows signs of increased organisation,” Jane's Intelligence Review, June 1, 2004.

[15] Rohan Gunaratna, “The asymmetric threat from maritime terrorism,” Jane's Fighting Ships, December 20, 2001.

[16] Yemen Gateway, “Attack on the USS Cole,” http://www.al-bab.com/yemen/cole1.htm dated December 12, 2001, last accessed October 8, 2007.

[17] Wikipedia, “USS Cole bombing,” http://en.wikipedia.org/wiki/USS_Cole_bombing last accessed October 8, 2007.

[18] Wikipedia, “Limburg (ship) bombing,” http://en.wikipedia.org/wiki/Limburg_%28ship%29_bombing last accessed October 8, 2007.

[19] Yemen Gateway, “Maritime Wars,” http://www.al-bab.com/yemen/artic/mei88.htm dated June 7, 2003, last accessed October 8, 2007.

[20] Terence Tan Kian Moh, “Naval Tactical Plan Generation for Littoral water Operation Using Conceptual Blending Theory,” M.S. thesis, Naval Postgraduate School, scheduled for submission December 2007.

[21] Ryan Tan Boon Leng, “A Study to Model Human Behavior in Discrete-event simulation (DES) using Simkit,” M.S. thesis, Naval Postgraduate School, scheduled for submission December 2007.

[22] Averill M. Law, “Simulation Modeling & Analysis,” 4th Ed., (McGraw-Hill series in industrial engineering and management science), Chapter One, pp. 6-79.

[23] Arnold H. Buss, “A Tutorial on Discrete-Event Modeling with Simulation Graphs,” Proceedings of the 1995 Winter Simulation Conference, pp. 74-81.

[24] Arnold H. Buss, “Modeling with Event Graphs,” Proceedings of the 1996 Winter Simulation Conference, pp. 153-160.

[25] Eric L. Savage, Lee W. Schruben, Enver Yücesan, “On the Generality of Event-Graph Models,” INFORMS Journal on Computing Vol. 17, No. 1, Winter 2005, pp. 3–9.

[26] Lee W. Schruben and Enver Yücesan, “Complexity of Simulation Models: A Graph Theoretic Approach,” Proceedings of the 1993 Winter Simulation Conference, pp. 641-649.

[27] Lee W. Schruben, “Building Reusable Simulators Using Hierarchical Event Graphs,” Proceedings of the 1995 Winter Simulation Conference, pp. 472-475.

[28] Arnold H. Buss, “Component-based Simulation Modeling,” Proceedings of the 2000 Winter Simulation Conference, pp. 964-971.

[29] Arnold H. Buss, Paul J. Sánchez, “Building Complex Models with LEGOs,” Proceedings of the 2002 Winter Simulation Conference, pp. 732-737.

[30] Arnold H. Buss, “Discrete Event Programming with Simkit,” Simulation New Europe Technical Notes, 2001.

[31] Arnold H. Buss, “Simkit Home Page,” http://diana.nps.edu/Simkit/, last accessed October 16, 2007.

[32] Koh Kim Leng, “A Study on Modeling Approaches in Discrete-event simulation using Design Patterns,” M.S. thesis, Naval Postgraduate School, scheduled for submission December 2007.

[33] GNU Project and Free Software Foundation, “Licenses – GNU GPL, GNU LGPL, GNU FDL, General Public License, Lesser General Public License, Free Documentation License, List of Free Software Licenses,” http://www.gnu.org/licenses/, last accessed October 16, 2007.

[34] Arnold H. Buss, “Component-based Simulation Modeling with Simkit,” Proceedings of the 2002 Winter Simulation Conference, pp. 243-249.

[35] Arnold H. Buss and Paul Sanchez, “Simple Movement and Detection in Discrete-event simulation,” Proceedings of the 2005 Winter Simulation Conference, pp. 992-1000.

[36] Rockwell Automation, Inc., “Rockwell – Arena Simulation,” http://www.arenasimulation.com/products/default.asp, last accessed October 16, 2007.

[37] Imagine That!, Inc, “Imagine That! – Extend Product Line,” http://www.imaginethatinc.com/prods_prodline.html, last accessed October 16, 2007.

[38] Brian Stout, “The Basics of A* for Path Planning,” Game Programming Gems, 2000.

[39] Chris Darken, “MV4025: Cognitive and Behavioral Modeling for Simulations” course notes, 2007.

[40] NOAA/National Geophysical Data Center, Marine Geology and Geophysics Division, “Coastline Extractor,” http://rimmer.ngdc.noaa.gov/mgg/coast/getcoast.html, last accessed August 21, 2007.

[41] Paul Tozour, “Search Space Representations,” AI Game Programming Wisdom II, pp. 85-102, 2003.

[42] Marco Pinter, “Toward More Realistic Pathfinding,” Gamasutra.com, 2001.

[43] Stan Franklin and Art Graesser, “Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents,” Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996.

[44] John Hiles, “MV4015: Agent-based autonomous behavior for simulations” course notes, 2007.

[45] Wikipedia, “Null hypothesis - Wikipedia,” http://en.wikipedia.org/wiki/Null_hypothesis, last accessed November 2, 2007.

[46] X. Hu, “Context-Dependent Adaptability in Crowd Behavior Simulation,” Proc. The 2006 IEEE International Conference on Information Reuse and Integration (IRI 2006), 2006.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Francis Lim Hup Leong
   Defence Science and Technology Agency
   Defence Technology Tower A
   Singapore

4. Mui Whye Kee
   Defence Science and Technology Agency
   Defence Technology Tower A
   Singapore

5. Arnold H. Buss
   Naval Postgraduate School
   Monterey, California

6. John Hiles
   Naval Postgraduate School
   Monterey, California

7. Chris Darkens
   Naval Postgraduate School
   Monterey, California

8. Tony Ciavarelli
   Naval Postgraduate School
   Monterey, California

9. Curtis Blais
   Naval Postgraduate School
   Monterey, California